SEE
FOLLOWING
NOTE

VOIR
NOTE
CI-APRÈS

PARTS OF THE ORIGINAL DOCUMENT

ARE OF POOR REPRODUCTIVE QUALITY

CERTAINES PARTIES DU DOCUMENT ORIGINAL

SONT DE PIÈTRE QUALITÉ DE REPRODUCTION

Environment Canada Environnement Canada

National Hydrology Research Institute

NHRI PAPER NO. 39
IWD TECHNICAL BULLETIN NO. 155

A Numerical Simulator for Flow and Transport in Stochastic Discrete Fracture Networks

A. Rouleau*

*Now at Département des Sciences Appliquées
Université du Québec à Chicoutimi
555 boulevard de l'Université
Chicoutimi, Québec G7H 2B1

NHRI

INLAND WATERS DIRECTORATE
NATIONAL HYDROLOGY RESEARCH INSTITUTE
NATIONAL HYDROLOGY RESEARCH CENTRE
SASKATOON, SASKATCHEWAN, 1988

(Disponible en français sur demande)

Published by authority of
the Minister of the Environment

© Minister of Supply and Services Canada 1988

Cat. No. En 36-503/155E

ISBN 0-662-16290-0

Contents

		Page
ABSTRACT		vi
RESUME		vi
ACKNOWLEDGMENTS		vii
1.	INTRODUCTION	1
2.	STOCHASTIC SIMULATION OF FRACTURE SYSTEMS: PROGRAMS NETWRK AND APEGEN	4
	2.1 Introduction	4
	2.2 Program NETWRK: Main program and input data	5
	2.3 Line pattern generation	7
	2.4 Generation of spacing data	10
	2.5 Line intersections	10
	2.6 Generation of a plotting file	12
	2.7 Determining the line elements	12
	2.8 A simple example	12
	2.9 Generation of supplementary aperture distributions: Program APEGEN	14
	2.10 Conclusion	17
3.	NUMERICAL SIMULATION OF FLUID FLOW IN A TWO-DIMENSIONAL DISCRETE FRACTURE NETWORK: PROGRAM NETFLO	18
	3.1 Introduction	18
	3.2 Model formulation	20
	3.3 Program NETFLO	22
	3.3.1 Preliminary operations	23
	3.3.2 Fluid-flow calculations	27
	3.4 Program verification and accuracy evaluation	31
	3.5 Conclusion	35
4.	STOCHASTIC SIMULATION OF PARTICLE TRANSPORT IN A FRACTURE NETWORK: PROGRAM NETRANS	36
	4.1 Introduction	36
	4.2 Program NETRANS	37
	4.2.1 Determining the flow direction	37
	4.2.2 Determining the average flow velocity in every direction	41
	4.3.2 Number of particles required	42
	4.3 Conclusion	42
5.	GENERAL CONCLUSION	44
REFERENCES		45
APPENDIX A.	LIST OF THE MAIN VARIABLES USED IN THE PROGRAMS NETWRK (NW), APEGEN (AP), NETFLO (NF), AND NETRANS (NT)	51
	A.1 List of input parameters	53
	A.2 List of parameters used internally	56

Contents (Cont.)

		Page
APPENDIX B.	INPUT/OUTPUT FILES	65
APPENDIX C.	FORMAT OF INPUT DATA	71
	C.1 General input file (Unit 1)	73
	C.2 Supplementary input file for the program APEGEN (Unit 21)	74
APPENDIX D.	CONVENTIONS FOR ANGLES, DIRECTIONS, AND BOUNDARY GEOMETRY	77
	D.1 Conventions for angles	79
	D.2 Conventions for the direction cosines computed in the program APEGEN	79
	D.3 Conventions and limitations on boundary geometry	81
APPENDIX E.	ARRAY DIMENSIONING	83
Appendix F.	SAMPLE RUNS	91
	F.1 A rectangular model without abutting fractures: Realization 990	93
	F.2 A rectangular model with abutting fractures: Realization 992	114
	F.3 A circular model: Realization 991	116
Appendix G.	PROGRAM LISTINGS	119
	G.1 Listing of the program NETWRK	121
	G.2 Listing of the program APEGEN	164
	G.3 Listing of the program NETFLO	169
	G.4 Listing of the program NETRANS	196

Tables

2.1	Selected results for the line network of Figure 2.7	15
B.1	List of all the input/output files	67
B.2	Input files and options of output files for each program	69

Illustrations

		Page
Figure 2.1.	Structure of program NETWRK	5
Figure 2.2.	Flowchart of the main program in NETWRK	6
Figure 2.3.	Model geometry: (A) rectangular model, (B) circular model	7
Figure 2.4.	Flowchart of subroutine GENLIN	8
Figure 2.5.	Flowchart of subroutine INTERS	11
Figure 2.6.	Flowchart of subroutine ELTDEF	13
Figure 2.7.	A simple line network generated by NETWRK	15
Figure 2.8.	Flowchart of the main program in APEGEN	16
Figure 3.1.	Fracture network consisting of five connected segments	20
Figure 3.2.	Structure of program NETFLO	22
Figure 3.3.	Flowchart of the main program in NETFLO	23
Figure 3.4.	Flowchart of subroutine OPSTOR	24
Figure 3.5.	A symmetric variable-bandwidth matrix	27
Figure 3.6.	Flowchart of subroutine SOLPHI	28
Figure 3.7.	Flowchart of subroutine FLOCAL	30
Figure 3.8.	Fracture system used for evaluating the accuracy of the program NETFLO	31
Figure 3.9.	Distribution of accumulated round-off error for various number of nodes between fixed-head boundaries in the system of Figure 3.8	33
Figure 3.10.	Relative error on mass balance as a function of number of nodes between fixed-head boundaries for the system of Figure 3.8	34
Figure 4.1.	Structure of program NETRANS	37
Figure 4.2.	Flowchart of the main program in NETRANS	38
Figure 4.3.	Set-up of the array IDIR (100) for the stochastic determination of the flow direction	39
Figure 4.4.	Flowchart of subroutine TTIMER	40
Figure 4.5.	Fracture system used for NETRANS verification	41
Figure 4.6.	Comparison of deterministic plug flow and the stochastic transit time of particles with different weighting factors for the flow velocity	42
Figure D.1.	Conventions for angles	79
Figure D.2.	Conventions for the direction cosines computed in the program APEGEN	80
Figure D.3.	Numbering of the flow boundaries: (A) rectangular model, (B) circular model	81
Figure F.1.	Boundary conditions for realization 990	93
Figure F.2.	Computer plot of realization 990	95
Figure F.3.	Rose diagram of file SUMFLO completed by hand	109
Figure F.4.	Computer plot of realization 992	115
Figure F.5.	Boundary conditions for realization 991	116
Figure F.6.	Computer plot for realization 991	118

Abstract

This report describes a stochastic discrete fracture (SDF) modelling package that simulates ground-water flow and mass transport in fracture systems. The program NETWRK generates two-dimensional fracture networks with a Monte Carlo method based on the statistics of field data on fracture geometry, i.e., fracture orientation, trace length, aperture, and density. The program APEGEN can be used to generate supplementary aperture distributions for the fracture network generated by NETWRK. A third program, NETFLO, computes the steady-state fluid flow through the fracture network generated by NETWRK and APEGEN. NETFLO also computes the statistics of selected parameters in every ten-degree range of direction, including the total length of fracture segments, the total flow velocity, and the total flow rate. The latter directional parameters are used by the program NETRANS to compute the transit time of particles over an arbitrary distance using a second-level stochastic process.

Résumé

Le rapport décrit un progiciel de modélisation de fractures discrètes stochastiques (FDS) qui simule l'écoulement de l'eau souterraine et le transfert des contaminants dans des réseaux de fractures. Le programme NETWRK génère des réseaux de fractures bidimensionnels à l'aide d'une méthode de Monte Carlo basée sur des valeurs recueillies sur le terrain des paramètres statistiques de configuration des fractures, c'est-à-dire l'orientation, la longueur des traces, l'ouverture et la densité des fractures. Le programme APEGEN permet de générer des distributions d'ouvertures supplémentaires pour le réseau de fractures généré par le programme NETWRK. Un troisième programme, NETFLO, calcule l'écoulement fluide en régime permanent dans le réseau de fractures généré par NETWRK et APEGEN. Il calcule aussi certains paramètres statistiques dans chaque intervalle de direction de dix degrés, notamment la longueur totale de segments de fractures, la vitesse d'écoulement totale et le débit total. Ces derniers paramètres directionnels sont utilisés par le programme NETRANS pour calculer le temps de transit des particules sur une distance arbitraire à l'aide d'une méthode stochastique de deuxième niveau.

Acknowledgments

Parts of these computer programs were developed for my Ph.D. thesis at the University of Waterloo. I benefited considerably from the numerous technical and scientific advice received from my thesis advisor, J.E. Gale, now at the Memorial University of Newfoundland. Many interesting discussions with K.S. Novakowski and K.G. Raven on transport processes in fractured rocks motivated the development of the program NETRANS. D. Lentz implemented much of the coding in NETRANS, and R. MacLeod contributed to the coding of APEGEN. Financial support for the development of the programs NETWRK and NETFLO came from three main sources: (1) a research grant to J.E. Gale, while at the University of Waterloo, from the Natural Sciences and Engineering Research Council of Canada; (2) scholarships from the F.C.A.C. funds of the Ministère de l'Education du Québec; and (3) a scholarship from the North American Life Insurance Company through the Canadian Council of Professional Engineers. Modifications to the original program NETWRK and the development of the program APEGEN were funded by Fracflow Consultants Inc. of St. John's, Newfoundland. The Inland Waters Directorate supported the development of the program NETRANS and the elaboration of the documentation of the entire SDF package.

Introduction

Hydrogeology in industrialized countries has witnessed a shift in interest during the last decade from ground-water quantity issues to ground-water quality and contaminant transport problems. For example, in the subject of ground-water flow into underground excavations, the focus of hydrogeological investigations was, in the past, essentially the total volume of water flowing into an excavation (e.g., Fawcett *et al.*, 1984). Similarly, in ground-water resource evaluation and in hydrocarbon reservoir engineering, the main concern has generally been the yield of geological formations. Hence, in these conventional engineering geology problems, the hydraulic parameters of interest have been principally the bulk permeability and storativity, since these parameters determine the available fluid flux and fluid volume under a given hydraulic gradient. Hydrogeological models that have been developed for the study of such problems in fractured rocks include models based on the continuum approach (Snow, 1969; Duguid and Lee, 1977) and models of discrete fractures (Wittke, 1970). The contribution of regular arrays of finite-size fractures to the permeability of a rock mass has been investigated with discrete fracture models (Huskey and Crawford, 1967; Prats, 1972; Asfari and Witherspoon, 1973). The discrete fracture approach has also been used to analyze the effect of distributed fracture conductances (Parsons, 1966) and distributed fracture sizes (Caldwell, 1972) on the permeability tensor of jointed rocks. Finally, the important effect of stress on fluid flow through fractures has been demonstrated in the field and in the laboratory, and coupled stress-flow numerical models have been used (Gale, 1977; Noorishad *et al.*, 1982).

Currently, much effort is devoted to the study of existing and potential contaminant migration from toxic waste disposal sites and radioactive waste repositories, particularly in fractured rock. Even though the permeability of a fractured rock mass is an important parameter in contaminant hydrogeology, other hydraulic parameters such as effective fracture porosity (Gale and Rouleau, 1984) and fluid velocity may be more important because they enable one to directly characterize the rate of migration of dissolved contaminants in ground water.

Clearly, the estimation of porosity and fluid velocity in fractured rocks requires a detailed description of the geometry of the fracture system. In general, rock fractures are grouped in sets on the basis of their orientation. Within a fracture set we observe variability in fracture orientation, spacing, and size. These distributed parameters must be characterized statistically (Baecher, 1983; Rouleau and Gale, 1985).

Similarly, a stochastic approach is appropriate for incorporating these geometric parameters in numerical models of fracture networks (Conrad and Jacquin, 1973; Bridges, 1975; Veneziano, 1978; Hudson and Priest, 1979; Baczynski, 1980; LaPointe and Hudson, 1985). Stochastic discrete fracture (SDF) models including a fluid-flow component have been developed first in two-dimensional domains (Hudson and LaPointe, 1980; Long *et al.*, 1982; Robinson, 1982a; Samaniego and Priest, 1984; Andersson *et al.*, 1984), and more recently with three-dimensional fracture networks (Dershowitz *et al.*, 1985; Long *et al.*, 1985). The movement of solutes through SDF networks of orthogonal fractures of distributed size and semi-random location has also been investigated (Schwartz *et al.*, 1983). Many of these references and others are further discussed in the appropriate part of the text.

Parametric studies, without reference to a particular field site, can indicate the theoretical relative effect of the various fracture-geometry parameters on selected dependent variables such as the total flow rate (Long *et al.*, 1982), the flow velocity, or the distribution of particle transit times (Smith and Schwartz, 1984). However, site-specific numerical simulations using real field data impose particular constraints on a study. Site-specific simulations require both detailed and accurate field data. Also, it is clear that realistic comparisons between field data and numerical model results can be made only if there is consistency in parameter definitions in the data and in the model.

The rationale for the SDF approach in a site-specific study was discussed by Gale and Rouleau (1984) within the framework of a detailed hydrogeological characterization of a waste repository in fractured rock with negligible matrix permeability. Rouleau and Gale (1987) reported an SDF simulation study of ground-water flow into an experimental drift excavated in the fractured granite at Stripa, Sweden.

This report documents the two programs used by Rouleau and Gale (1987), NETWRK and NETFLO, and two other related programs, APEGEN and NETRANS. The program NETWRK uses a Monte Carlo approach to generate two-dimensional discrete fracture networks based on the statistics of the main fracture-geometry parameters: fracture density, orientation, trace length, and aperture. The program APEGEN can then be used to generate different aperture distributions using the same fracture network geometry generated by NETWRK. The program NETFLO computes the steady-state fluid flow in the generated network, subject to specific boundary conditions; it also computes cumulative values of a number of flow parameters in each ten-degree range of direction. These latter directional flow parameters can then be used by the program NETRANS, which again uses a Monte Carlo approach to compute the transit time of particles over an arbitrary distance (Rouleau, 1987).

I have adopted a modular approach for the development of the programs. This makes the package more flexible in case of further additions or modifications. Also, it is possible to analyze the results of one of the programs before running another program on the same network realization.

The programs have been designed for site-specific applications. For instance, the input data on fracture geometry can be measured directly in the field (Rouleau and Gale, 1985). In other SDF simulation programs, the number of fractures has generally been used as one of the input parameters (Long *et al.*, 1982; Robinson, 1982a; Schwartz *et al.*, 1983; Samaniego and Priest, 1984; Smith and Schwarz, 1984). Here we replace the number of fractures by fracture density (total length of fracture traces per unit surface area) as an input parameter because borehole data generally provide a relatively good estimate of this latter parameter, independently of fracture size.

Other particularities of this S F simulation package include the optimization of the node numbering, the linear array storage of the node conductance matrix, and the second-level stochastic process invoked for the particle transport.

Stochastic Simulation of Fracture Systems: Programs NETWRK and APEGEN

2.1 Introduction

During the last decade, a number of workers have used numerical techniques to simulate fracture systems in rock masses. Conrad and Jacquin (1973) proposed the generation of two-dimensional (2D) line patterns to study the geometry and the size distribution of rock blocks. Hudson and Priest (1979) presented a three-dimensional (3D) approach to the same problem. Bridges (1975) proposed a 3D model based on random discs in space in combination with 2D fracture trace plots to "convey a mental image of fracture patterns." Veneziano (1978) presented another 3D model based on Poisson polygons on random planes. Veneziano's model included the generation of random infinite planes in space. He then used random lines in each one of the planes to define convex polygons, some of which represented discontinuous fractures. The same author discussed possible applications of this model in assessing the mechanical properties of a rock mass. In an effort at modelling the spatial variability of fracturing, Baczynski (1980) generated 2D line patterns aimed at reproducing, for a given fracture set, the log-normal spacing distribution observed in the field. La Pointe and Hudson (1985) also presented numerically generated 2D line patterns using various types of distributions for line orientation and line length. More recently, Long *et al.* (1982), Robinson (1982b), and Samaniego and Priest (1984) presented the results of 2D random line network generations combined with the numerical calculation of steady-state fluid-flow parameters (see Chap. 3). Using a numerical approach based on the percolation theory developed in the field of theoretical physics, Robinson (1982a) found critical densities for 2D random fracture networks below which no continuous flowpath exists in a fracture system.

The line network generation code NETWRK presented in this chapter uses a Monte Carlo method to generate a pattern of lines of distributed lengths and orientations in a fashion similar to the last four studies mentioned above. In these models, each line represents the trace of a fracture that cuts a slice of rock one unit in thickness. The lines in NETWRK are generated one set at a time, using separate distribution parameters for each set for the lengths and the orientations. Fracture apertures are also selected from a given distribution and assigned to each line. Output files are thus created that are directly usable by other programs, for instance, to calculate flow in 2D fracture networks.

A 2D model constitutes a first step in the numerical simulation of discrete fracture networks. Such a model can be used to

evaluate the effects of fractures on the mechanical or the flow properties of a rock mass. Even though a 3D model could give a more realistic representation of a fracture system, no numerical 3D model is available yet for large systems (i.e., thousands) of discrete fractures distributed in size and orientation. Moreover, a 3D model would require considerably more computer storage and computing time than a 2D model for the same number of fractures simulated.

2.2 Program NETWRK: Main Program and Input Data

The structure of program NETWRK is shown in Figure 2.1. This program executes the following sequence of operations (Fig. 2.2): (1) reading of input data, (2) generation of a line pattern, (3) computation of spacing values (optional), (4) location of all the "effective" intersections in the network (i.e., those intersections that are part of a continuous flow path), (5) generation of a plotting file (optional), and (6) definition of the "elements" (i.e., every line segment between two consecutive effective intersections) and recording of the node numbers that identify each element.

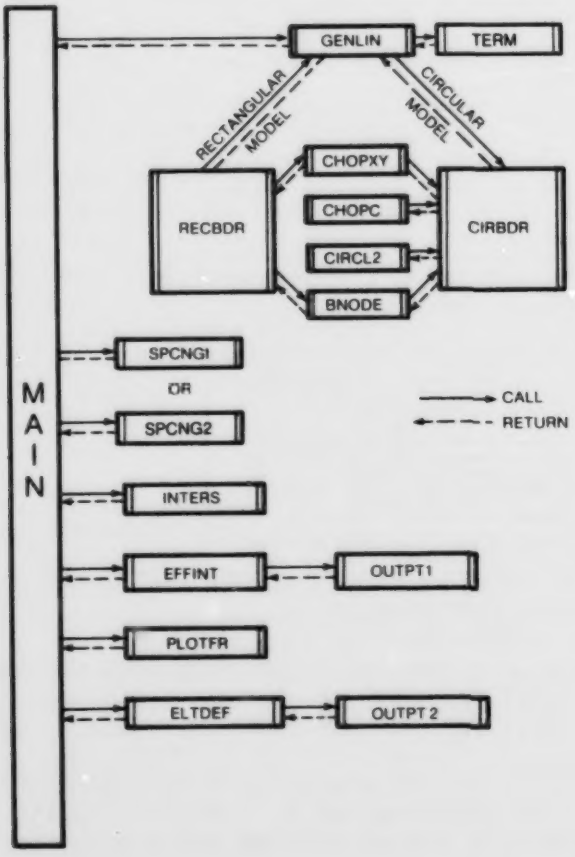


Figure 2.1. Structure of program NETWRK.

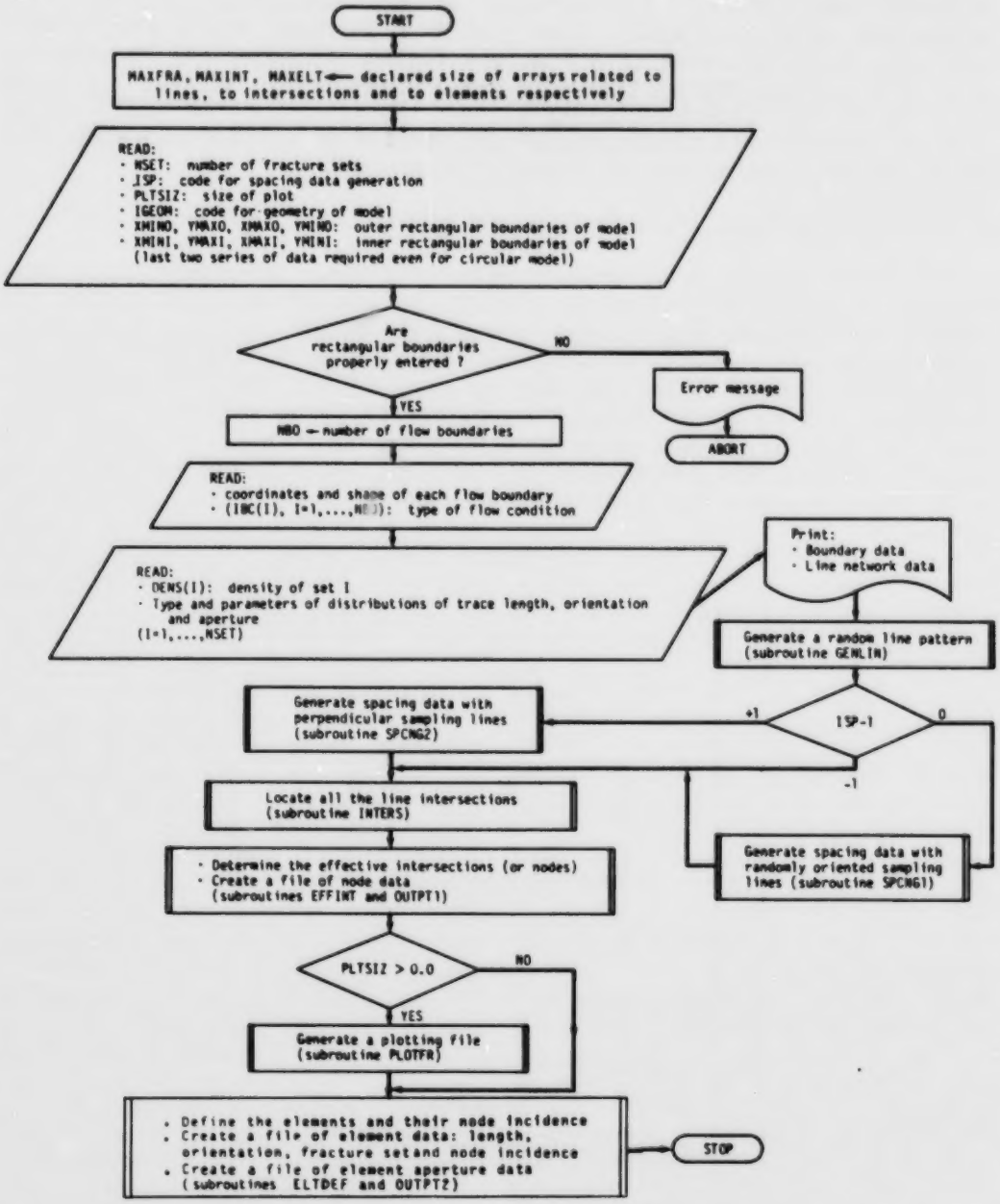

START
MAXFRA, MAXINT, MAXELT ← declared size of arrays related to lines, to intersections and to elements respectively
READ:
· NSET: number of fracture sets
· ISP: code for spacing data generation
· PLTSIZ: size of plot
· IGEOM: code for geometry of model
· XMINO, YMAXO, XMAXO, YMINO: outer rectangular boundaries of model
· XMINI, YMAXI, XMAXI, YMINI: inner rectangular boundaries of model
(last two series of data required even for circular model)
Are rectangular boundaries properly entered ?
NO
YES
Error message
ABORT
NBO ← number of flow boundaries
READ:
· coordinates and shape of each flow boundary
· (IBC(I), I=1,...,NBO): type of flow condition
READ:
· DENS(I): density of set I
· Type and parameters of distributions of trace length, orientation and aperture
(I=1,...,NSET)
Print:
· Boundary data
· Line network data
Generate a random line pattern (subroutine GENLIN)
ISP-1
+1
0
-1
Generate spacing data with perpendicular sampling lines (subroutine SPCNG2)
Generate spacing data with randomly oriented sampling lines (subroutine SPCNG1)
Locate all the line intersections (subroutine INTERS)
· Determine the effective intersections (or nodes)
· Create a file of node data
(subroutines EFFINT and OUTPT1)
PLTSIZ > 0.0
NO
YES
Generate a plotting file (subroutine PLOTFR)
· Define the elements and their node incidence
· Create a file of element data: length, orientation, fracture set and node incidence
· Create a file of element aperture data
(subroutines ELTDEF and OUTPT2)
STOP

Figure 2.2. Flowchart of the main program in NETWRK.

Input data for the program are primarily of two types: (1) data describing the geometry of the fracture pattern and (2) data related to the boundary geometry (shape and coordinates). The fracture pattern is defined by the density of each fracture set (DENS [L^{-1}]) and the type and the parameters of the distribution of fracture trace

lengths, orientations, and apertures for each fracture set. NETWRK can handle two types of model geometry: rectangular models and circular models (Fig. 2.3; see also Appendix D). In a circular model, only one quadrant of a complete annulus is represented, defined by two concentric circles and two radial lines. Each model needs both an outer boundary and an inner boundary. The lines of the network are generated over the entire model inside the outer boundaries. The inner boundaries are used to trim the network on all the sides, and they are the only boundaries considered in the program NETFLO (Chap. 3). The buffer margin between the inner and the outer boundaries eliminates most of the edge effect that a generated line network acquires by the fractures being systematically less dense near the boundaries.

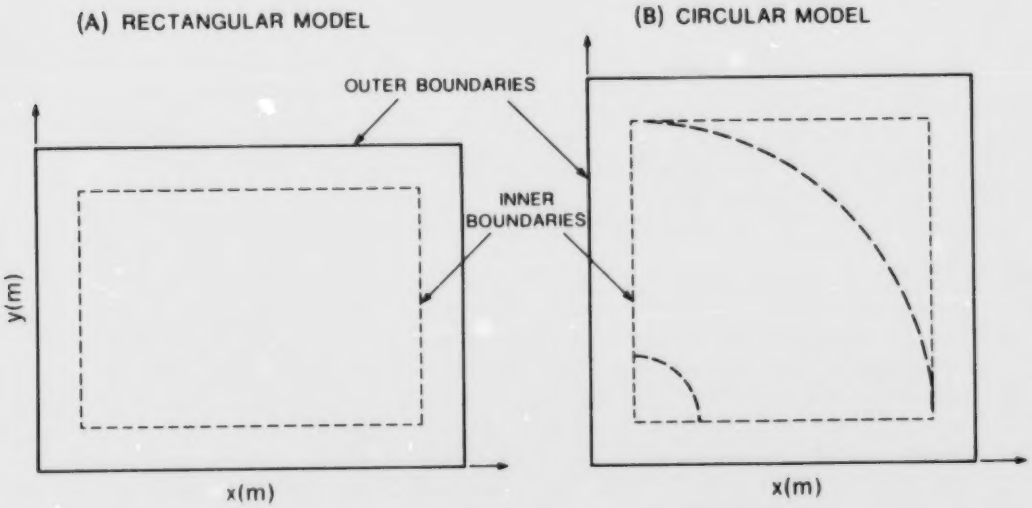


Figure 2.3. Model geometry: (A) rectangular model, (B) circular model.

2.3 Line Pattern Generation

The generation of the line pattern is carried out by the subroutine GENLIN (Fig. 2.4). In this step, the computer first calculates the surface area of the model inside the inner boundaries. The line pattern is then generated one set at a time. The desired total length (DESLEN) for the lines is calculated as well as an estimate of the number of lines to be generated (ESTNFR). The variables DESLEN and ESTNFR are used as control parameters in the line generation process. The value of ESTNFR is set at a specified factor COEF (an input parameter typically between 2 and 5) times the initial estimate of the number of lines based on mean trace length and the value of DESLEN. This value of ESTNFR is greater than what is generally needed, but it ensures that the desired fracture density will

be reached in all network realizations. In some cases, the extra lines that are generated just compensate for a high proportion of short traces appearing at the beginning of the trace length generation process, and for a high number of traces located in the margin between the outer and the inner boundary as explained below.

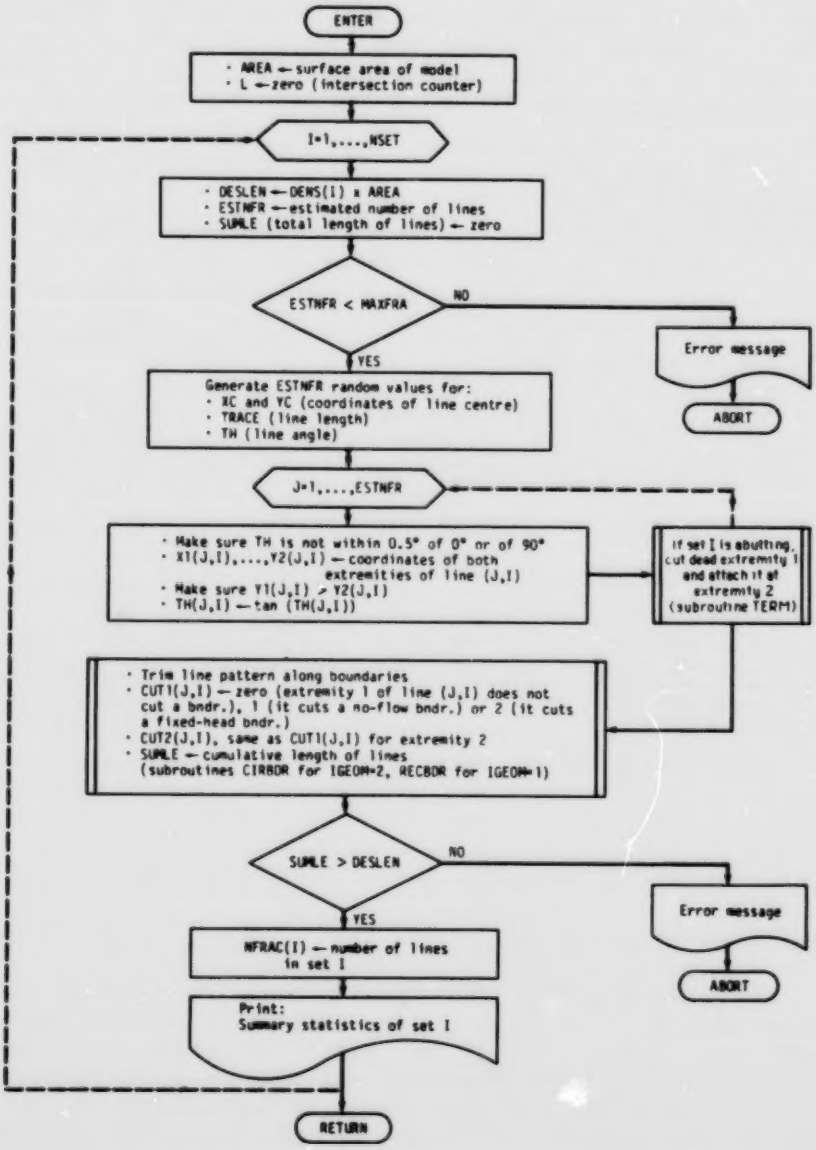

ENTER
· AREA ← surface area of model
· L ← zero (intersection counter)
I=1,...,NSET
· DESLEN ← DENS(I) x AREA
· ESTNFR ← estimated number of lines
· SUMLE (total length of lines) ← zero
ESTNFR < MAXFRA
NO
Error message
ABORT
YES
Generate ESTNFR random values for:
· XC and YC (coordinates of line centre)
· TRACE (line length)
· TH (line angle)
J=1,...,ESTNFR
· Make sure TH is not within 0.5° of 0° or of 90°
· X1(J,I),...,Y2(J,I) ← coordinates of both extremities of line (J,I)
· Make sure Y1(J,I) > Y2(J,I)
· TH(J,I) ← tan (TH(J,I))
If set I is abutting, cut dead extremity 1 and attach it at extremity 2 (subroutine TERM)
· Trim line pattern along boundaries
· CUT1(J,I) ← zero (extremity 1 of line (J,I) does not cut a bndr.), 1 (it cuts a no-flow bndr.) or 2 (it cuts a fixed-head bndr.)
· CUT2(J,I), same as CUT1(J,I) for extremity 2
· SUMLE ← cumulative length of lines
(subroutines CIRBDR for IGEOM=2, RECBDR for IGEOM=1)
SUMLE > DESLEN
NO
Error message
ABORT
YES
NFRAC(I) ← number of lines in set I
Print:
Summary statistics of set I
RETURN

Figure 2.4. Flowchart of subroutine GENLIN.

Next, a fixed number ESTNFR of random points are generated inside the outer boundary, each one corresponding to the centre of a

line (XC, YC). Then the subroutine generates the same number of (1) line lengths (TRACE, in metres), using either a single-value or a log-normal distribution; and (2) line orientations (TH, in degrees), using either a single-value or a normal distribution. The elements of the arrays XC, YC, TRACE, and TH are used concurrently later in the program to calculate the coordinates of both extremities of each line (X1, Y1 and X2, Y2).

Concerning the computation of the line extremities, two procedures are taken to facilitate the operations in other subroutines. First, all line angles within 0.5 degree from 0.0, 90.0, 180.0, or 270.0 are set to either -0.5, +0.5, 89.5, or 90.5. For the lines sub-parallel to the x-axis (Fig. 2.3), this procedure ensures that no indeterminate value will occur in the definition of the line elements (Sec. 2.7). Indeed, the elements along a line are defined after sorting (by increasing y-value) all the intersections along that line. In the case of lines sub-parallel to the y-axis (Fig. 2.3), this same procedure puts an upper limit to the value of the tangent of the line angles. It thus prevents floating point errors, particularly in the computation of the line intersections (Sec. 2.5). The second procedure simply ensures that extremity 1 (i.e., X1, Y1) is the extremity with the higher y-value. This convention makes the programming easier, in particular for the line trimming along the inner boundaries, which is briefly described below.

At this point, subroutine TERM is optionally called (if a value of 1 is given to the input parameter ITERM) to abut the fractures of the set being generated against fractures of previously generated sets. This procedure is intended to simulate more realistically natural systems that are often made of abutting fractures. Subroutine TERM cuts the end segment above the highest (with respect to the y-axis) intersection along a fracture; it then attaches this segment at the lower end of the same fracture so that the total length of the fracture stays the same. In this manner, only one end of a given fracture is abutting. Also the fractures of a given set can abut only on fractures of the sets that were generated previously. Obviously, fractures of set 1 cannot abut on other fractures. An example of this abutment procedure is given in realization 992 described in Appendix F.

The line pattern is trimmed along the inner boundaries by either subroutine RECBDR for a rectangular model, or subroutine CIRBDR for a circular model. These two subroutines, and other ancillary subprograms called by them, reject any line completely outside the inner boundaries, cut all the lines crossing an inner boundary, compute a corrected length for these lines, and compute the new coordinates of the trimmed extremities. Every intersection of a line and an inner boundary is also defined as a boundary node. The coordinates of each node are recorded in the arrays XORD and YORD, while the array IB identifies the boundary on which a node is located. Subroutines RECBDR and CIRBDR also sum up progressively for each set the total length (SUMLE) of the line segments inside the inner boundaries. These subroutines return execution to the line generator GENLIN when the variable SUMLE reaches the desired total length (DESLEN) for that set. The number of lines considered for each set is then recorded in NFRAC.

2.4 Generation of Spacing Data

As can be seen in the flowchart of the main program (Fig. 2.2), after the line generation step for all the sets, the code provides the option of generating a spacing data file. The number of spacing values generated for each fracture set is arbitrarily set at 150. Spacing is defined here as the distance between two consecutive intersections of a sampling line with fractures of the same set multiplied by the cosine of the angle between the sampling line and the pole of the average plane for that set. The user is given the choice between two subroutines for spacing data generation depending on the type of sampling process that is desired. The subprogram SPCNG1 uses only sampling lines perpendicular to the average direction of a set. SPCNG2 uses sampling lines randomly oriented in the plane of the model. The statistical analysis of the generated spacing data can be used for comparison with the field spacing data (e.g., Rouleau and Gale, 1985).

2.5 Line Intersections

After all the lines have been generated, the main program calls the subroutines INTERS and EFFINT to locate all the effective line intersections in the network.

The point of intersection between two lines in a plane, when one point and the orientation are known for each line, can be calculated using simple trigonometric relationships. To calculate the ordinate first, one can use:

$$YI = \frac{Y1(1)}{\tan TH(1)} - \frac{Y1(2)}{\tan TH(2)} - X1(1) + X1(2) \times \frac{\tan TH(1) \times \tan TH(2)}{\tan TH(2) - \tan TH(1)} \quad (2.1)$$

where YI is the ordinate of the intersection point, X1 (1 and 2) and Y1 (1 and 2) are the coordinates of a known point on lines 1 and 2 (e.g., extremity 1 of the lines on the network), and TH (1 and 2) is the angle between the lines (1 and 2) and the x-axis. The abscissa of the intersection point, XI, can then be calculated using the relationship:

$$XI = \frac{YI - Y1(1)}{\tan TH(1)} + X1(1) \quad (2.2)$$

The subroutine INTERS (Fig. 2.5) consists basically of four nested DO-loops. For each individual line, the computer scans all the other lines in the network, but measures are taken to ensure that the intersection between the same two lines is sought only once. First, the ordinate (YI) of a possible intersection between two lines is computed using Equation 2.1. If this value of YI is within the range of both lines in the y-direction, then the intersection does exist and its abscissa is computed using (2.2). The arrays XORD and YORD record the intersection coordinates. The number of the lines forming an intersection are recorded in the arrays INC1 and INC2.

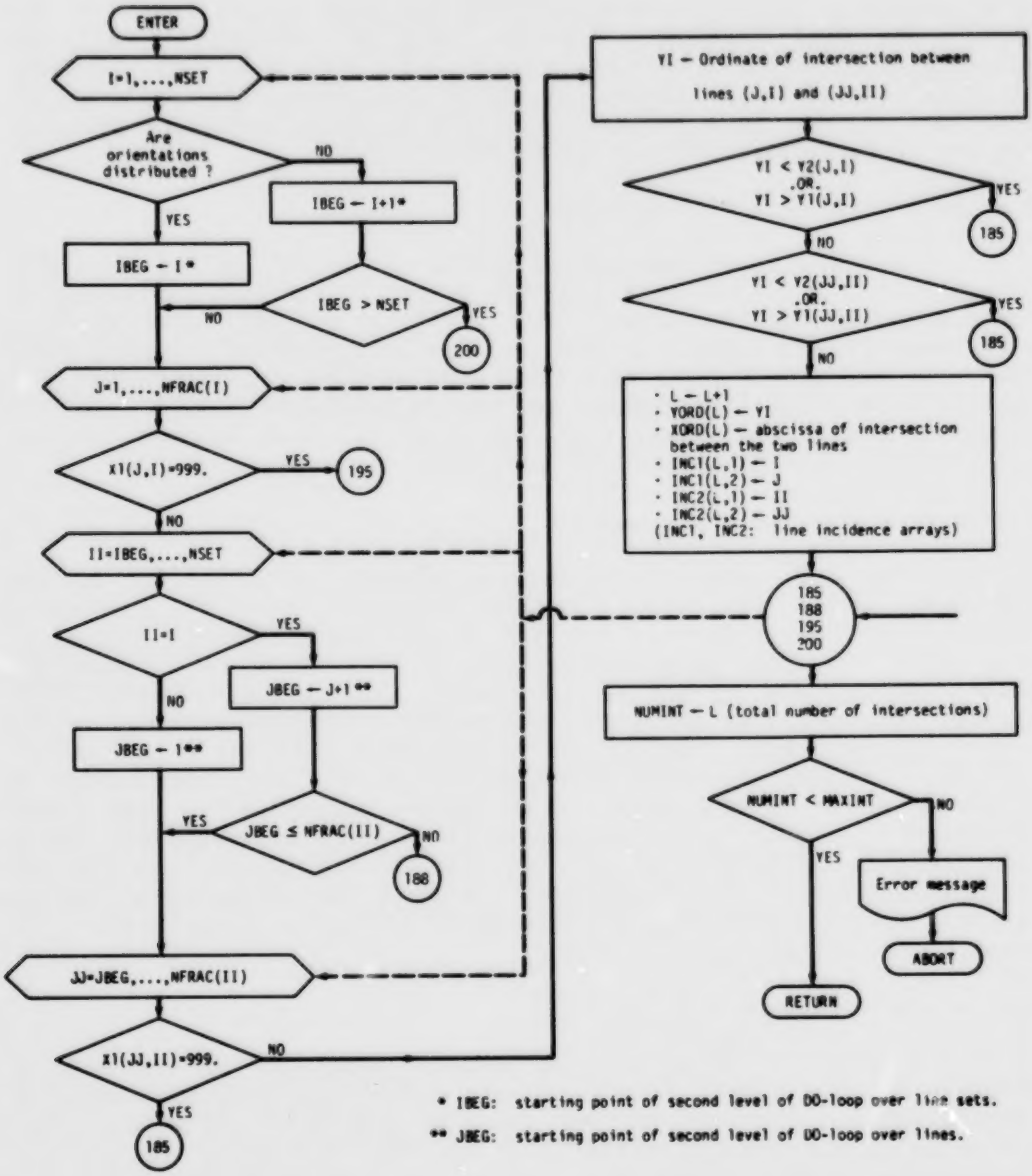


Figure 2.5. Flowchart of subroutine INTERS.

After subroutine INTERS has returned the execution to the main program, the subroutine EFFINT is called to tag most of the non-effective intersections, i.e., those intersections not located along a continuous flowpath between two fixed-head boundaries. These non-effective intersections are identified by tagging systematically all intersections where one of the intersecting lines has only this one

intersection. Effective intersections are also called nodes in the text. One must remember, however, that not necessarily all the non-effective intersections are tagged with this procedure. Indeed isolated closed-loops formed by three or more intersections are sometimes present in a network, in which case our procedure does not tag any of these intersections as non-effective.

2.6 Generation of a Plotting File

At this stage, subroutine PLOTFR is called (unless the input parameter PLTSIZ is set to a negative value) to create a plotting file for drawing of the line network with a digital plotter. The resulting plot shows all the boundaries and the line pattern after trimming along the inner boundaries. The effective intersections are identified by a small circle. The line numbers and the intersection numbers are also shown on a plot with fewer than 200 effective intersections, if the value of the input parameter IPRT is greater than 1.

2.7 Determining the Line Elements

The last step in the program NETWRK is carried out by the subroutine ELTDEF (Fig. 2.6). In three nested DO-loops, the subroutine ELTDEF first identifies all the effective intersections on any given line using the line incidence arrays INC1 and INC2. After ordering these intersections by increasing y-value, all the sub-segments thus defined are identified and their intersection incidence recorded in the arrays NOD1 and NOD2. During the same process, the subroutine ELTDEF assigns a random "fracture aperture" to every line, from either a single-value or a log-normal distribution according to the input parameters. ELTDEF also eliminates the dead-end segments, which comprise both complete lines and line extremities, after recording the total length and total "volume" for each set. The subroutine ELTDEF is repeated for each different aperture distribution for a particular line network realization.

2.8 A Simple Example

Figure 2.7 shows a line network generated by the program NETWRK. For this simple example, single values have been used for line lengths and orientations and for fracture apertures. Note also that the value of the input parameter IPRT was one; therefore, the line and intersection numbers are not written on the computer plot for Figure 2.7. Morever, the line termination control parameter ITERM was set to zero for both sets; so no line abuts on other lines. A selection of the information contained in the output for this example is given in Table 2.1. Other items optionally provided in output, but not shown in Table 2.1, include (1) a list of all the lines in each set, with their length, their orientation, and the coordinates of their extremities; (2) a list of all the intersections with their coordinates and the numbers of the intersecting lines; and (3) a list of all the elements with the node incidence numbers, the numbers of the lines they belong to, their length, and their corresponding fracture aperture. Appendix B gives a list of the different output files and Appendix F provides more detailed examples.

The simple example of Figure 2.7 already points out an interesting aspect of a network of discontinuous fractures: for the same fracture density, the proportion of dead fracture segments is higher for a set with shorter fractures (75% for set 2 in this example) than for a set with longer fractures (34% for set 1).

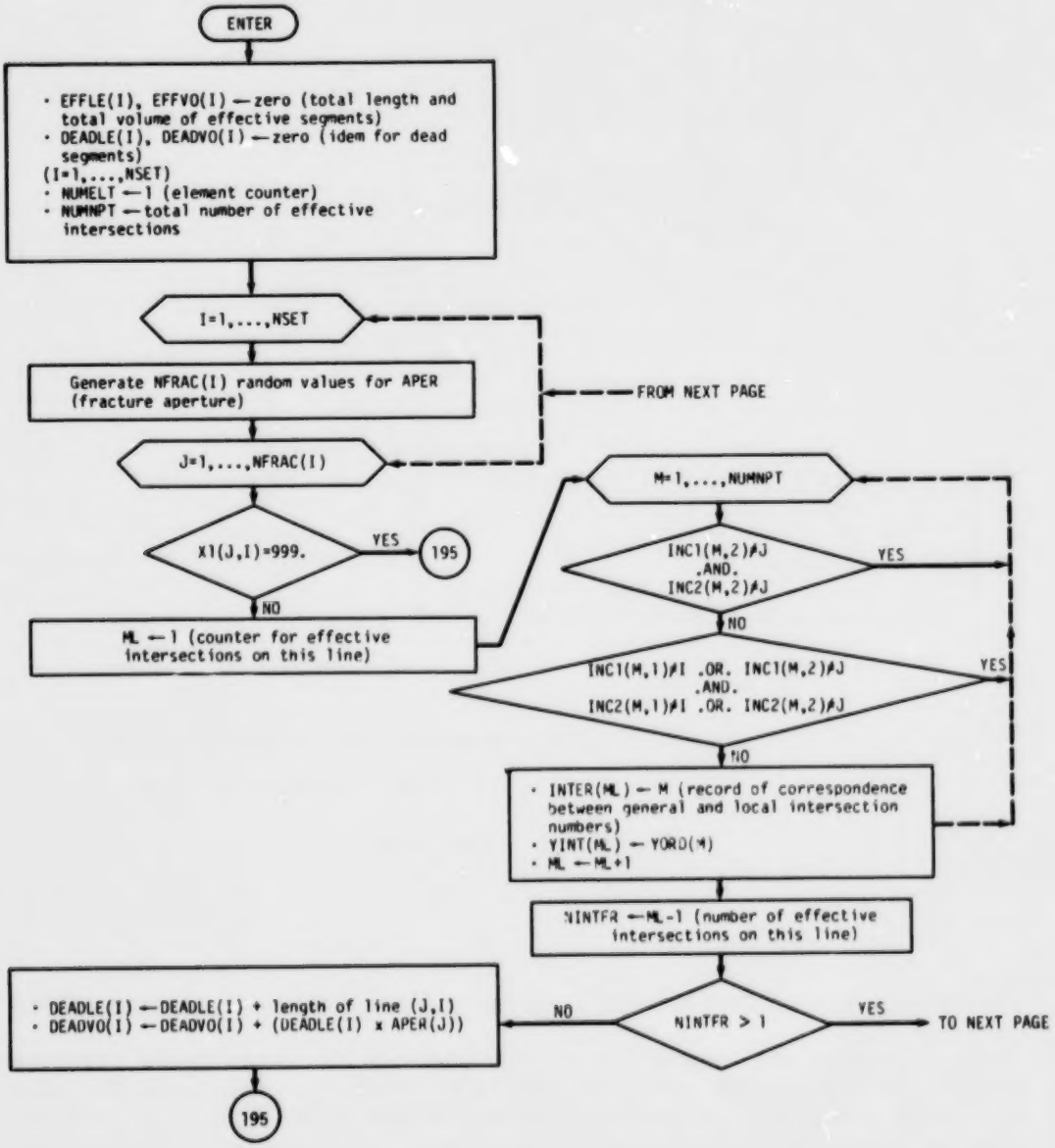

ENTER
· EFFLE(I), EFFVO(I) ← zero (total length and total volume of effective segments)
· DEADLE(I), DEADVO(I) ← zero (idem for dead segments)
(I=1,...,NSET)
· NUMELT ← 1 (element counter)
· NUMNPT ← total number of effective intersections
I=1,...,NSET
Generate NFRAC(I) random values for APER (fracture aperture)
FROM NEXT PAGE
J=1,...,NFRAC(I)
X1(J,I)=999.
YES
195
NO
ML ← 1 (counter for effective intersections on this line)
M=1,...,NUMNPT
INC1(M,2)≠J .AND. INC2(M,2)≠J
YES
NO
INC1(M,1)≠I .OR. INC1(M,2)≠J .AND. INC2(M,1)≠I .OR. INC2(M,2)≠J
YES
NO
· INTER(ML) ← M (record of correspondence between general and local intersection numbers)
· YINT(ML) ← YORD(M)
· ML ← ML+1
NINTER ← ML-1 (number of effective intersections on this line)
NINTER > 1
YES
TO NEXT PAGE
NO
· DEADLE(I) ← DEADLE(I) + length of line (J,I)
· DEADVO(I) ← DEADVO(I) + (DEADLE(I) x APER(J))
195

Figure 2.6. Flowchart of subroutine ELTDEF.

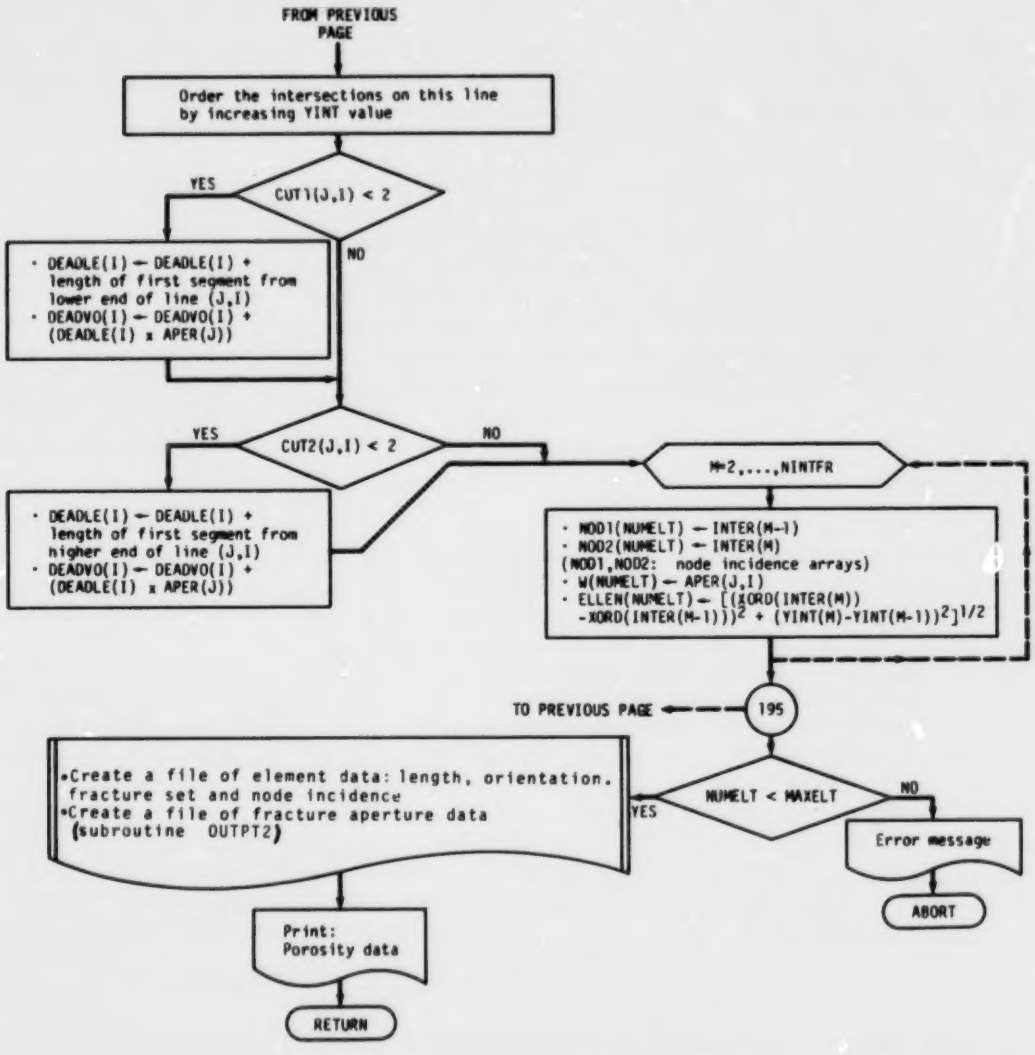

FROM PREVIOUS PAGE
Order the intersections on this line by increasing YINT value
CUT1(J,I) < 2
YES
NO
• DEADLE(I) ← DEADLE(I) + length of first segment from lower end of line (J,I)
• DEADVO(I) ← DEADVO(I) + (DEADLE(I) x APER(J))
CUT2(J,I) < 2
YES
NO
• DEADLE(I) ← DEADLE(I) + length of first segment from higher end of line (J,I)
• DEADVO(I) ← DEADVO(I) + (DEADLE(I) x APER(J))
M=2,...,NINTFR
• NOD1(NUMELT) ← INTER(M-1)
• NOD2(NUMELT) ← INTER(M)
(NOD1,NOD2: node incidence arrays)
• W(NUMELT) ← APER(J,I)
• ELLEN(NUMELT) ← [(XORD(INTER(M)) -XORD(INTER(M-1)))2 + (YINT(M)-YINT(M-1))2]1/2
TO PREVIOUS PAGE
195
NUMELT < MAXELT
YES
NO
Error message
ABORT
•Create a file of element data: length, orientation, fracture set and node incidence
•Create a file of fracture aperture data (subroutine OUTPT2)
Print: Porosity data
RETURN

Figure 2.6. Continued.

2.9 Generation of Supplementary Aperture Distributions: Program APEGEN

Within the subroutine ELTDEF (Sec. 2.7), distributed aperture values are generated and an aperture value is assigned to each generated fracture. A second program, APEGEN (Fig. 2.8), can also be used to generate different aperture values. The program APEGEN allows the same generated network to be used as many times as desired and with different aperture distribution parameters.

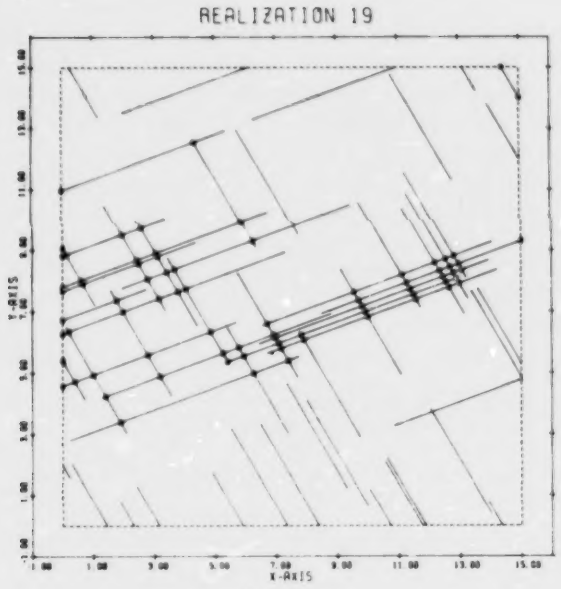


Figure 2.7. A simple line network generated by NETWRK.

Table 2.1. Selected Results for Line Network of Figure 2.7

Total number of intersections (NUMINT):	97	
Number of effective intersections (NUMNPT):	76	
Number of elements (NUMELT):	108	
	Set 1	Set 2
DESLEN (m)	112.5	112.5
SUMLE (m)	114.8	113.8
NFRAC	18	39
EFFLE (m)	76.4	28.4
DEADLE (m)	38.4	85.5
EFFVO (m^3)	1.5×10^{-3}	5.7×10^{-4}
DEADVO (m^3)	7.7×10^{-4}	1.7×10^{-3}

An additional benefit of APEGEN is that different elements along the same fractures are given different aperture values. Finally, with only minor additions, APEGEN could compute aperture values that are functions of the orientation or the location of the fracture elements.

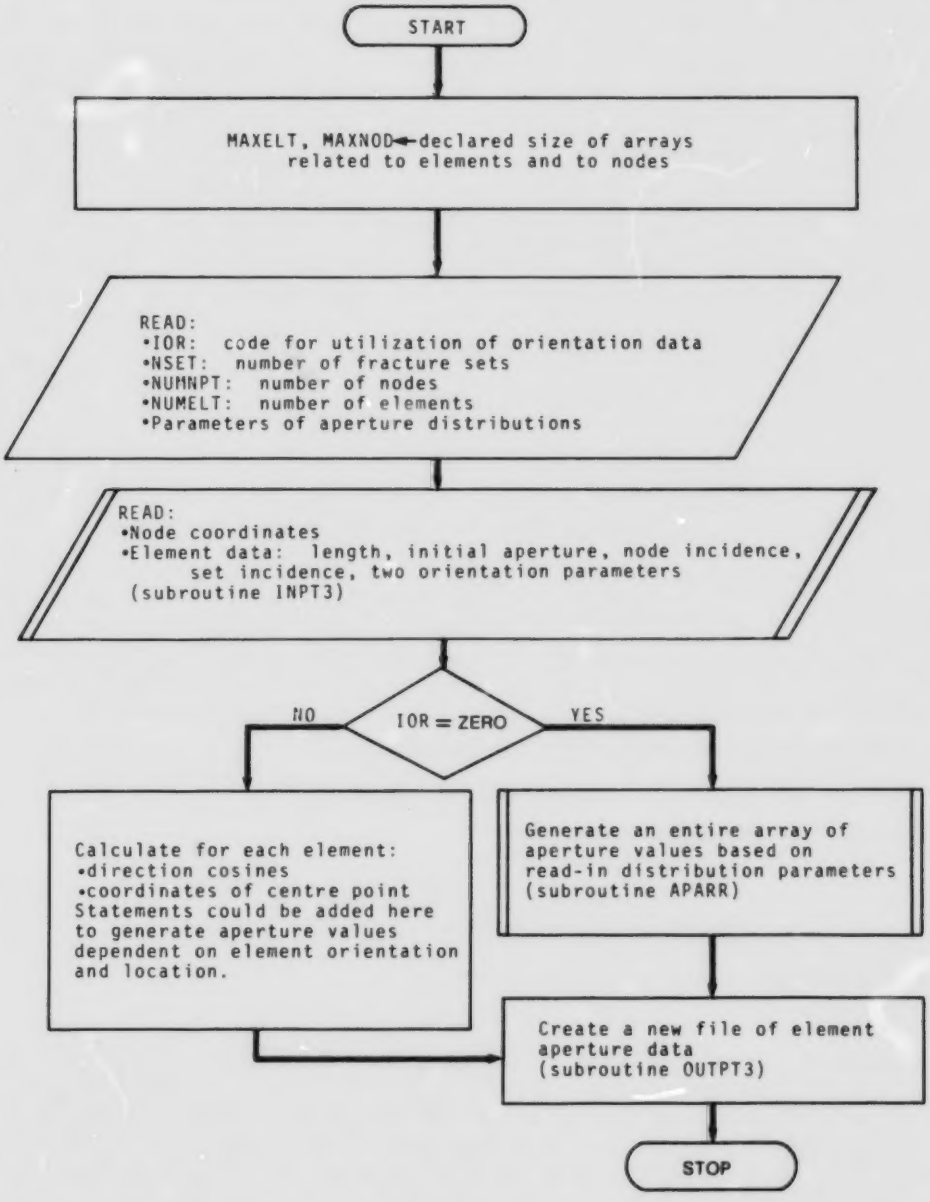

START
MAXELT, MAXNOD←declared size of arrays related to elements and to nodes
READ:
•IOR: code for utilization of orientation data
•NSET: number of fracture sets
•NUMNPT: number of nodes
•NUMELT: number of elements
•Parameters of aperture distributions
READ:
•Node coordinates
•Element data: length, initial aperture, node incidence, set incidence, two orientation parameters
(subroutine INPT3)
NO
IOR = ZERO
YES
Calculate for each element:
•direction cosines
•coordinates of centre point
Statements could be added here to generate aperture values dependent on element orientation and location.
Generate an entire array of aperture values based on read-in distribution parameters
(subroutine APARR)
Create a new file of element aperture data
(subroutine OUTPT3)
STOP

Figure 2.8. Flowchart of the main program in APEGEN.

2.10 Conclusion

Program NETWRK is a numerical tool that can be used in a number of applications. By itself, it provides information on the topology of a fracture system (e.g., degree of interconnection, ratio of effective to total porosity, and spacing distribution). NETWRK can be used in conjunction with other computer programs to evaluate the mechanical behaviour of a fractured rock mass or to compute the parameters of fluid flow through a fracture network. The following chapter describes the program NETFLO, which computes the steady-state fluid-flow parameters through a fracture network generated by NETWRK, given specific boundary conditions.

Numerical Simulation of Fluid Flow in a Two-dimensional Discrete Fracture Network: Program NETFLO

3.1 Introduction

This chapter presents the numerical model NETFLO, which simulates the flow of fluid in discrete fracture networks like those generated by the program NETWRK (Chap. 2). The program NETFLO has been developed to evaluate the role of fracture geometry in determining the ground-water flow properties of rock masses with a low-permeability matrix.

Previous studies have proposed a variety of techniques to simulate fluid flow in discrete fractures. Ollos (1963) constructed a physical model formed by PVC pipes interconnected at right angles to study the ground-water flow characteristics of a fracture system. Huskey and Crawford (1967) described an electric analog model consisting of an electrolyte bath and copper strips simulating fractures. Wittke (1970) used another electric analog composed of many electrical resistors arranged in networks. Caldwell (1972) used a third type of electric analog consisting of conducting paper with perforations to simulate impermeable rock blocks. Caldwell's technique is the inverse image of a technique described by Marcus and Evenson (1961) to simulate the effect of regular arrangements of thin impermeable layers in a permeable matrix. Hudson and LaPointe (1980) proposed an electric analog model consisting of a printed circuit that was a replica of a computer-generated random joint trace pattern. Numerical methods also developed for electric resistance network problems have been adapted to study fluid flow in fracture networks (Parsons, 1966; and Louis, 1969). Castillo *et al.* (1972) used a combination of an iterative solution and a direct solution to solve the steady-state flow problem in fracture networks of regular geometry. With this latter approach, it is possible to consider non-linear problems introduced by fracture roughness and turbulent flow.

The finite element method has been used by Wittke (1970) to solve a steady-state, three-dimensional fluid-flow problem through a network of continuous fractures in an impermeable matrix. Wilson and Witherspoon (1970) developed a two-dimensional finite element model to solve the same fluid-flow problem through a fracture network of arbitrary geometry. Wilson and Witherspoon's code is efficient in computing time, but not in its use of computer memory, since it requires the storage of a complete square matrix of order N, N being

the number of nodes. They presented examples of fully connected networks of regular geometry. Asfari and Witherspoon (1973) presented the results of another finite element program applied to completely non-connected regular patterns of short fractures in a permeable matrix. Long *et al.* (1982) adapted Wilson and Witherspoon's code to randomly generated two-dimensional networks of discontinuous fractures in an impermeable matrix. Also working on random networks, Robinson (1982b) used a direct solution (Gaussian elimination) to solve the steady-state flow problem, while Samaniego and Priest (1984) tried two different methods: an iterative method (relaxation) and a direct method (bi-factorization). More recently, numerical models have been developed to simulate steady-state fluid flow in three-dimensional fracture networks. Dershowitz *et al.* (1985) used a finite-element approach; Long *et al.* (1985) adopted a mixed analytical-numerical technique based on line sources (sink); while Shapiro and Andersson (1985) formulated a boundary-element approach. In theory, a three-dimensional (3D) model of flow in a fracture system is more realistic than a two-dimensional (2D) model. Unfortunately, no suitable 3D model is available yet that would simulate flow in networks of the order of thousands of fracture intersections. Also, using the same size of computer memory and similar software, one can simulate a much larger fracture system in a 2D model than with a 3D model.

The two-dimensional fracture flow theory used here is very similar to the theory of fluid flow in pipe networks. This similarity was used by Ollos (1963) in physical models. A considerable amount of literature is devoted to the numerical solution of flow in pipe networks carrying, for instance, water, natural gas, or petroleum products (Mah and Shacham, 1978). Typically a pipe network problem has a predetermined general structure and numerical methods are used for sensitivity analysis and design optimization. Since the same problem structure is repeated many times during a study, it is profitable to optimize right at the beginning the structure of the matrices to be solved numerically. Linear theory and graph theory methods are commonly used for these types of pipe network analysis (Kesavan and Chandrashekar, 1972; Isaacs and Mills, 1980).

The numerical model presented here is two-dimensional, and the rock matrix is assumed impermeable. A node conductance matrix is first set up by applying the mass balance constraint to each node in the network (i.e., the sum of flow rates at any free node must equal zero). Since this computer code is intended to be applied to actual field problems, it must simulate the fractures in a slice of rock large enough to be considered statistically homogeneous with respect to the fracture system. Moreover, the random fracture generation process produces a numerical matrix that is very dispersed in structure. By renumbering systematically all the nodes and by using a variable-bandwidth storage scheme, the code presented here reduces considerably the computer storage requirement for a large sparse matrix. A direct method of solution based on the Choleski algorithm is then used to solve the set of equations in a manner that is efficient in computing time.

3.2 Model Formulation

For steady-state laminar flow of an incompressible, viscous fluid between two smooth parallel plates, the analytical solution of the Navier-Stokes equation yields the following relationship for volume flow rate per unit width (e.g., Huitt, 1956):

$$q = -\frac{W^3\gamma}{12\mu}\frac{\Delta\varphi}{\Delta x} \tag{3.1}$$

where W is the plate separation [L], γ is the weight density of the fluid [F/L^3], μ is the dynamic viscosity [FT/L^2], x is the distance along the plates [L], and φ is the hydraulic head [L]. In the solution of network flow problems, it is often convenient to define the conductance of the fluid conduit of length ℓ as:

$$e = \frac{W^3\gamma}{12\mu\,\ell} \qquad [L^2\,T^{-1}] \tag{3.2}$$

i.e., the hydraulic conductivity multiplied by the cross-section and divided by the length.

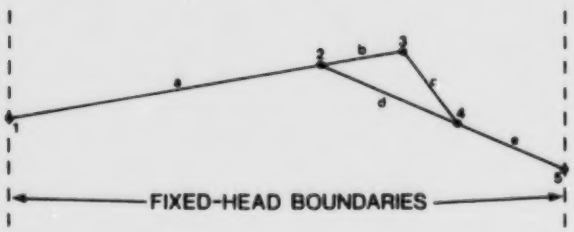


Figure 3.1. Fracture network consisting of five connected segments.

Using Figure 3.1 as an example, the flow rate at node 1 is expressed by:

$$e_a(\varphi_2-\varphi_1) = -q_a \tag{3.3}$$

In this convention, flow into the system is positive and flow out is negative. For the internal node 2, the mass balance constraint requires that the sum of flow rates into and out of that node equals zero:

$$e_a(\varphi_1-\varphi_2) + e_b(\varphi_3-\varphi_2) + e_d(\varphi_4-\varphi_2) = 0 \tag{3.4}$$

Written in matrix form, the equations (3.3) and (3.4) for the complete system of Figure 3.1 give:

$$\begin{array}{l} \text{nodes} \rightarrow \\ \downarrow \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{pmatrix} e_{-a} & -e_a & 0 & 0 & 0 \\ -e_a & (e_a+e_b+e_d) & -e_b & -e_d & 0 \\ 0 & -e_b & (e_b+e_c) & -e_c & 0 \\ 0 & -e_d & -e_c & (e_c+e_d+e_a) & -e_e \\ 0 & 0 & 0 & -e_e & e_e \end{pmatrix} \end{array} \times \begin{pmatrix} \varphi_1 \\ \varphi_2 \\ \varphi_3 \\ \varphi_4 \\ \varphi_5 \end{pmatrix} = \begin{pmatrix} q_1 \\ 0 \\ 0 \\ 0 \\ q_5 \end{pmatrix} \quad (3.5)$$

The numerous methods of simultaneously solving systems of equations like (3.5) can be divided in direct methods and iterative methods (e.g., Jennings, 1977). Direct methods are generally faster but require larger computer memory than iterative methods. For instance, Wilson and Witherspoon (1970) used a matrix reordering scheme by which all the unknowns (e.g., φ_2, φ_3, φ_4, q_1, and q_5 in Equation 3.5) are put in the same vector and then solved all at once with a direct method. Their code requires the complete storage of the square matrix.

In the code NETFLO, the matrix and vectors of Equation 3.5 are rearranged and partitioned according to the degree of freedom of the nodes:

$$\begin{array}{l} \text{nodes} \rightarrow \\ \downarrow \\ 2 \\ 3 \\ 4 \\ 1 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccc|cc} 2 & 3 & 4 & 1 & 5 \end{array} \\ \left(\begin{array}{ccc|cc} (e_a+e_b+e_d) & -e_b & -e_d & -e_a & 0 \\ -e_b & (e_b+e_c) & -e_c & 0 & 0 \\ -e_d & -e_c & (e_c+e_d+e_e) & 0 & -e_e \\ \hline -e_a & 0 & 0 & e_a & 0 \\ 0 & 0 & -e_e & 0 & e_e \end{array}\right) \end{array} \times \begin{pmatrix} \varphi_2 \\ \varphi_3 \\ \varphi_4 \\ \varphi_1 \\ \varphi_5 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ q_1 \\ q_5 \end{pmatrix} \quad (3.6)$$

This matrix equation can be summarized by

$$\begin{pmatrix} E_{ff} & E_{fc} \\ E_{cf} & E_{cc} \end{pmatrix} \times \begin{pmatrix} \Phi_f \\ \Phi_c \end{pmatrix} = \begin{pmatrix} Q_f \\ Q_c \end{pmatrix} \quad (3.7)$$

This corresponds to two matrix equations:

$$E_{ff}\Phi_f + E_{fc}\Phi_c = Q_f \quad (3.8)$$

and

$$E_{cf}\Phi_f + E_{cc}\Phi_c = Q_c \quad (3.9)$$

Equation 3.8 can be used to compute the unknown head values (ϕ_f), whereas (3.9) can give the flow rate at the boundary nodes (Q_c). The approach taken in program NETFLO is to compute the head at all the free nodes using (3.8) and then to compute the flow rate in all the segments individually using Equation 3.1. Rearranging (3.8) gives:

$$E_{ff}\phi_f = Q_f - E_{fc}\phi_c = Q_f' \qquad (3.10)$$

After setup of the augmented flux vector Q_f', program NETFLO solves for ϕ_f using the Choleski algorithm (e.g., Jennings, 1977, p. 107).

3.3 Program NETFLO

The program NETFLO has been written to execute the mathematical operations described in the preceding section. The structure of this program is shown in Figure 3.2. Sequentially, NETFLO (Fig. 3.3) (1) reads in the boundary conditions and the line network data generated by NETWRK, (2) defines the boundary condition for every boundary node, (3) renumbers all the nodes to reduce the bandwidth of matrix E and determines the codes for matrix condensation and storage, (4) solves for unknown head values, and (5) computes the flow rate in every segment, the total flow at every boundary, and the cumulative flow parameters in every ten-degree range of direction.

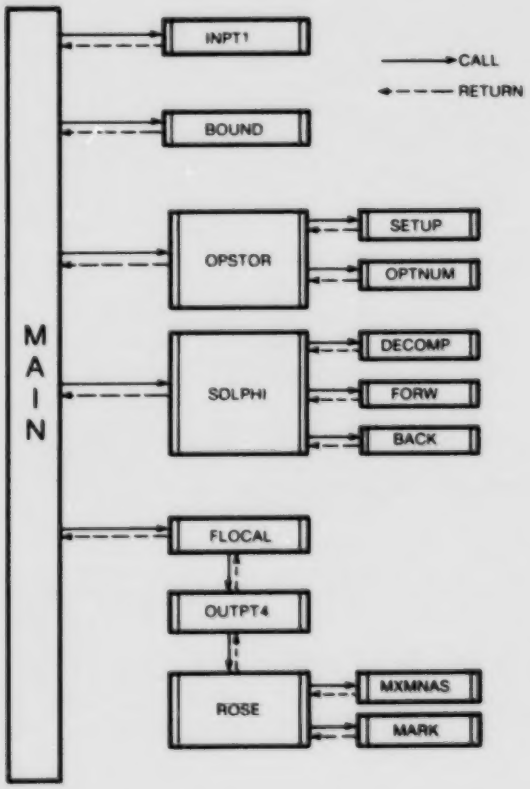


Figure 3.2. Structure of program NETFLO.

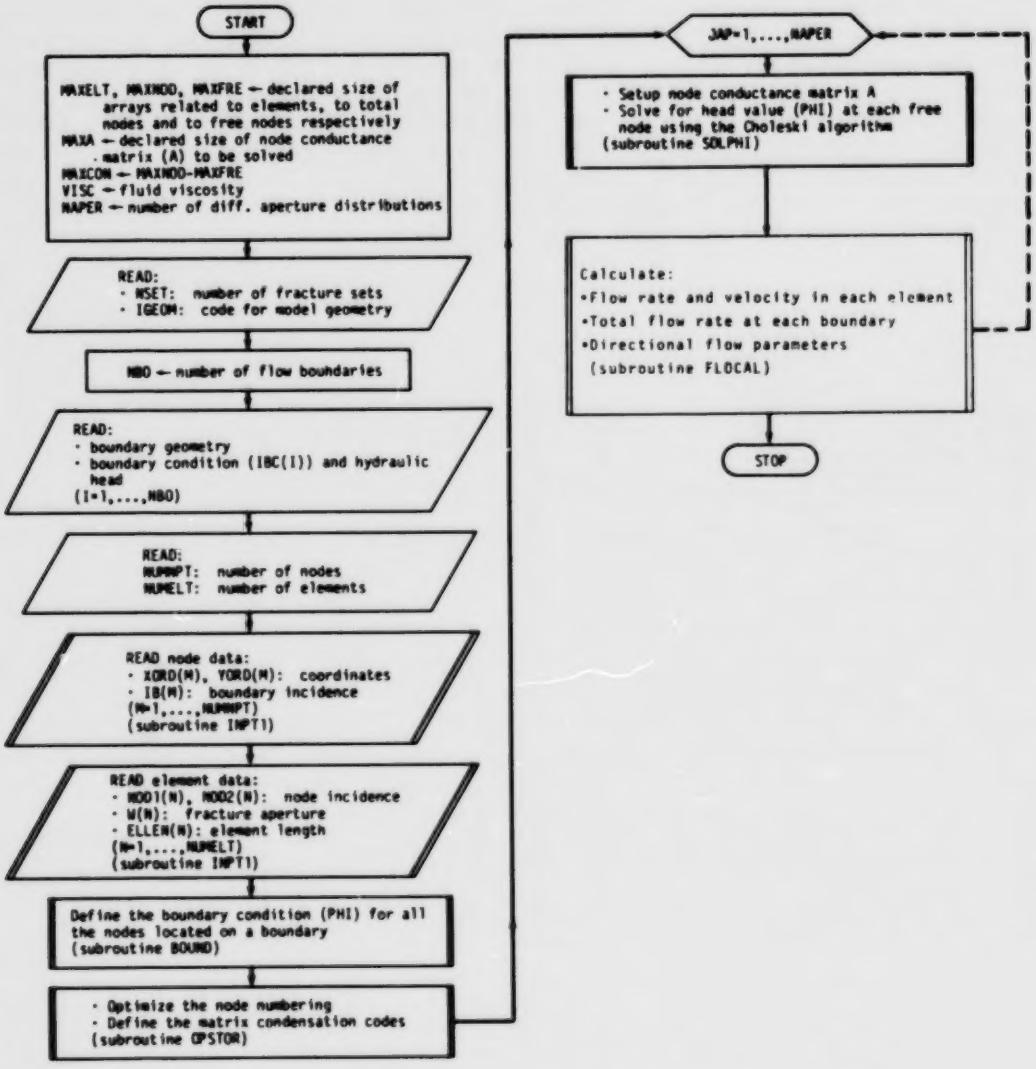

START
MAXELT, MAXNOD, MAXFRE ← declared size of arrays related to elements, to total nodes and to free nodes respectively
MAXA ← declared size of node conductance matrix (A) to be solved
MAXCOM ← MAXNOD-MAXFRE
VISC ← fluid viscosity
NAPER ← number of diff. aperture distributions
READ:
· NSET: number of fracture sets
· IGEOM: code for model geometry
NBO ← number of flow boundaries
READ:
· boundary geometry
· boundary condition (IBC(I)) and hydraulic head
(I=1,...,NBO)
READ:
NUMNPT: number of nodes
NUMELT: number of elements
READ node data:
· XORD(N), YORD(N): coordinates
· IB(N): boundary incidence
(N=1,...,NUMNPT)
(subroutine INPT1)
READ element data:
· NOD1(N), NOD2(N): node incidence
· W(N): fracture aperture
· ELLEN(N): element length
(N=1,...,NUMELT)
(subroutine INPT1)
Define the boundary condition (PHI) for all the nodes located on a boundary
(subroutine BOUND)
· Optimize the node numbering
· Define the matrix condensation codes
(subroutine OPSTOR)
JAP=1,...,NAPER
· Setup node conductance matrix A
· Solve for head value (PHI) at each free node using the Choleski algorithm
(subroutine SOLPHI)
Calculate:
•Flow rate and velocity in each element
•Total flow rate at each boundary
•Directional flow parameters
(subroutine FLOCAL)
STOP

Figure 3.3. Flowchart of the main program in NETFLO.

3.3.1 Preliminary Operations

Since NETFLO is applied to large problems in terms of number of fractures and number of intersections, the computer storage requirement becomes a critical factor. For that reason, a maximum size is defined in declaration statements in the main program for all the variables requiring a large array. Also, the same maximum values are assigned to the variables MAXELT, MAXNOD, MAXFRE, and MAXA, which are passed on to the various subroutines by a COMMON statement. This system allows an easy adjustment of the array size when needed.

NETFLO reads in first the data concerning the boundary geometry and flow conditions. With respect to fluid flow, a boundary can be specified as either a no-flow boundary (IBC=0) or a fixed-head boundary (IBC>0). Then NETFLO reads in the intersection (or node) data generated by NETWRK, i.e., the intersection coordinates (XORD, YORD) and the boundary incidence (IB) for all the intersections of a line with a boundary. Finally, the element data are read in. These are simply the node incidence (NOD1 and NOD2), and the length, the orientation, and the aperture of each element.

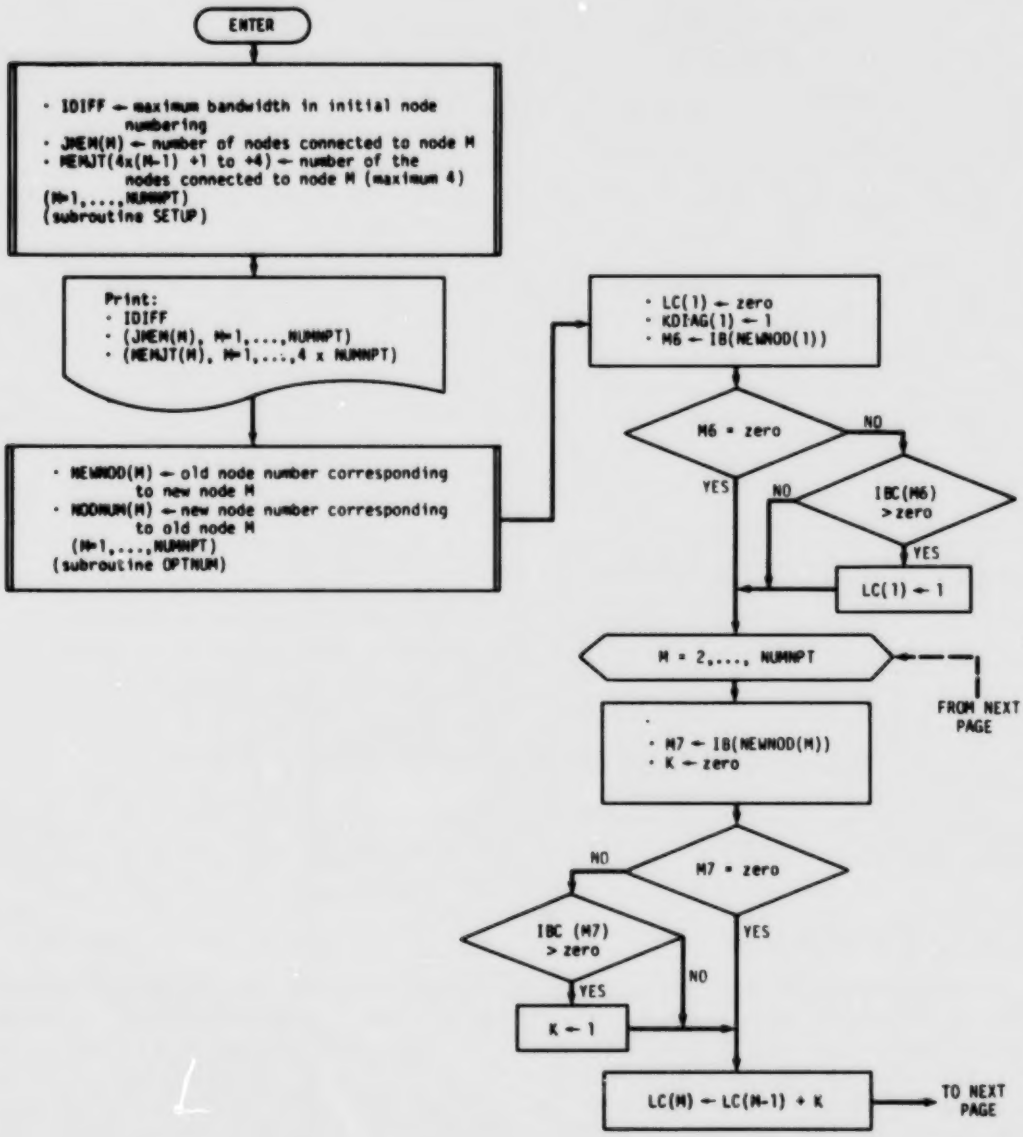


Figure 3.4. Flowchart of subroutine OPSTOR.

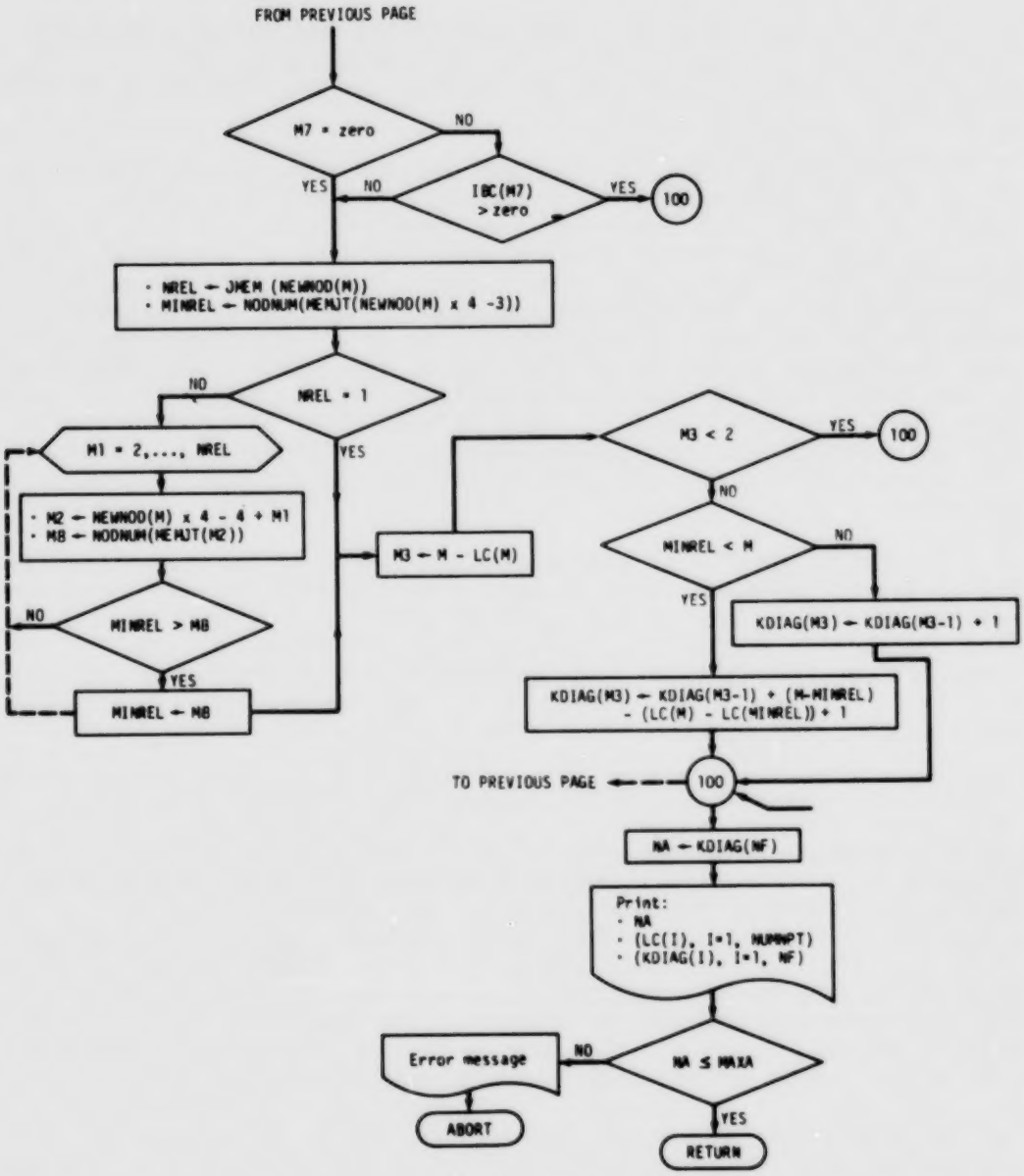


Figure 3.4. Continued.

The first computational step in NETFLO is to assign the exact value of head (PHI) to every node located on a fixed-head boundary. This step is carried out by subroutine BOUND, which considers whether the head is constant along a boundary (IBC=1), linearly varying (IBC=2), or logarithmically varying (IBC=3).

In order to apply NETFLO to problems that include a large number of fractures, two measures are taken to increase the storage

efficiency of this program: node renumbering and variable-bandwidth, one-dimensional array storage of the main flow matrix. This optimization step is carried out mainly by the subroutine OPSTOR (Fig. 3.4).

Because of the random process used for line network generation (Chap. 2), the node numbers are essentially randomly distributed in the model. In this initial numbering, two connected nodes can have a large difference in node numbers, which would produce a non-banded, very dispersed flow matrix A. For that reason, OPSTOR renumbers systematically all the nodes in the model, reducing considerably the bandwidth of the matrix A that is set up in a subsequent step. The node renumbering scheme is carried out by two subroutines, SETUP and OPTNUM, based on a code written by Collins (1973). Modifications have been added to Collins' code to make it applicable to cases of two or more unconnected but continuous flowpaths. In these cases, separate and unconnected pathways can carry the fluid from one model boundary to the other.

OPSTOR (Fig. 3.4) also determines the value of the entries in the arrays LC and KDIAG, both of which are used later in addressing the entries of the various arrays used in the flow calculations. LC is used essentially for matrix partitioning: the value of LC(I) is the number of constrained nodes that have an index smaller than or equal to I. In the example of Figure 3.1, the values of LC for the corresponding node number (index) are:

$$LC(1) = LC(2) = LC(3) = LC(4) = 1 \text{ and } LC(5) = 2$$

KDIAG contains the address of the diagonal entries in matrix A computed in a subsequent step. A is a symmetric, diagonally dominant matrix corresponding to the submatrix E_{ff} of Equation 3.7. It is stored in a variable-bandwidth, one-dimensional array (see Jennings, 1977, p. 97). Figure 3.5 shows a 6 x 6 symmetric matrix that illustrates the variable bandwidth storage method. If we store only the lower triangular matrix of Figure 3.5, the number of elements to be kept in memory for the six rows are 1, 1, 2, 1, 5, and 4 respectively. This matrix can be stored in the one-dimensional array A:

$$\begin{array}{cccccccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 \end{array}$$

$$A = \{a_{11} | a_{22} | a_{32}\ a_{33} | a_{44} | a_{51}\ 0\ a_{53}\ a_{54}\ a_{55} | a_{63}\ a_{64}\ 0\ a_{66}\ \} \qquad (3.11)$$

In order to interpret the array 3.11, the address of the six diagonal elements is specified in the integer array:

$$KDIAG = \{1\ 2\ 4\ 5\ 10\ 14\} \qquad (3.12)$$

Variable-bandwidth storage is suitable for matrix A since, even after the node renumbering carried out previously, the irregular structure of the line pattern produces many re-entrants in the resulting band matrix. A regular band storage scheme would require the storage of more zero elements. At the other extreme, the complete elimination of all the zero elements can also be done, for instance, by

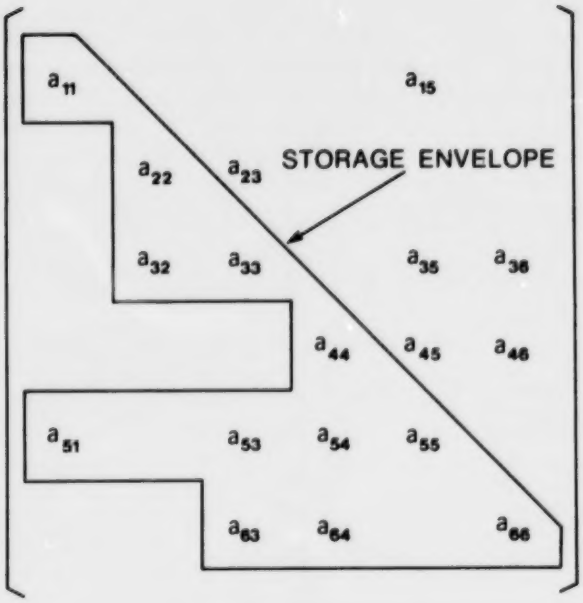


Figure 3.5. A symmetric variable-bandwidth matrix.

using two integer arrays of element addresses. The complete packing of array A, however, necessitates a more complex computer code, and moreover, it may not offer a considerable saving in memory storage because of the extra address array required.

3.3.2 Fluid-flow Calculations

In the following steps, the program NETFLO (Fig. 3.3) sets up the arrays for fluid-flow computation using the storage scheme defined previously, and solves the system for the unknown head values. Both operations are executed by the subroutine SOLPHI (Fig. 3.6).

The value of the expression $\gamma/12\mu$ (see Equation 3.1) is assigned to the constant FLUID at the beginning of the subroutine. Then for each fracture segment, the element conductance E is computed according to Equation 3.2 as:

$$e_N = W_N^3 \times \text{FLUID}/\ell_N \qquad (3.13)$$

where N is the element index, W is the fracture aperture, and ℓ is the element length. Clearly, each one of the two nodes connected by the element N is either of free type (unknown head) or of constrained type (fixed head). If at least one of the nodes is of free type, the value of e is entered in the appropriate location in array A. For a constrained node, e is multiplied by the corresponding head value, and the result is entered directly into the augmented flux vector (Q_f in Equation 3.10). This latter vector is stored in the array PHI.

The matrix system is then solved for all the unknown head values using three short subroutines (DECOMP, FORW, and BACK) adapted from Jennings (1977). DECOMP executes a Choleski decomposition of the matrix A (see Jennings, 1977, p. 107). FORW and BACK execute, respectively, a forward substitution and a back substitution in the system of equations. The array PHI is used again to store the solution in each one of these last two operations. This multiple use of the array PHI represents a significant saving in computer memory.

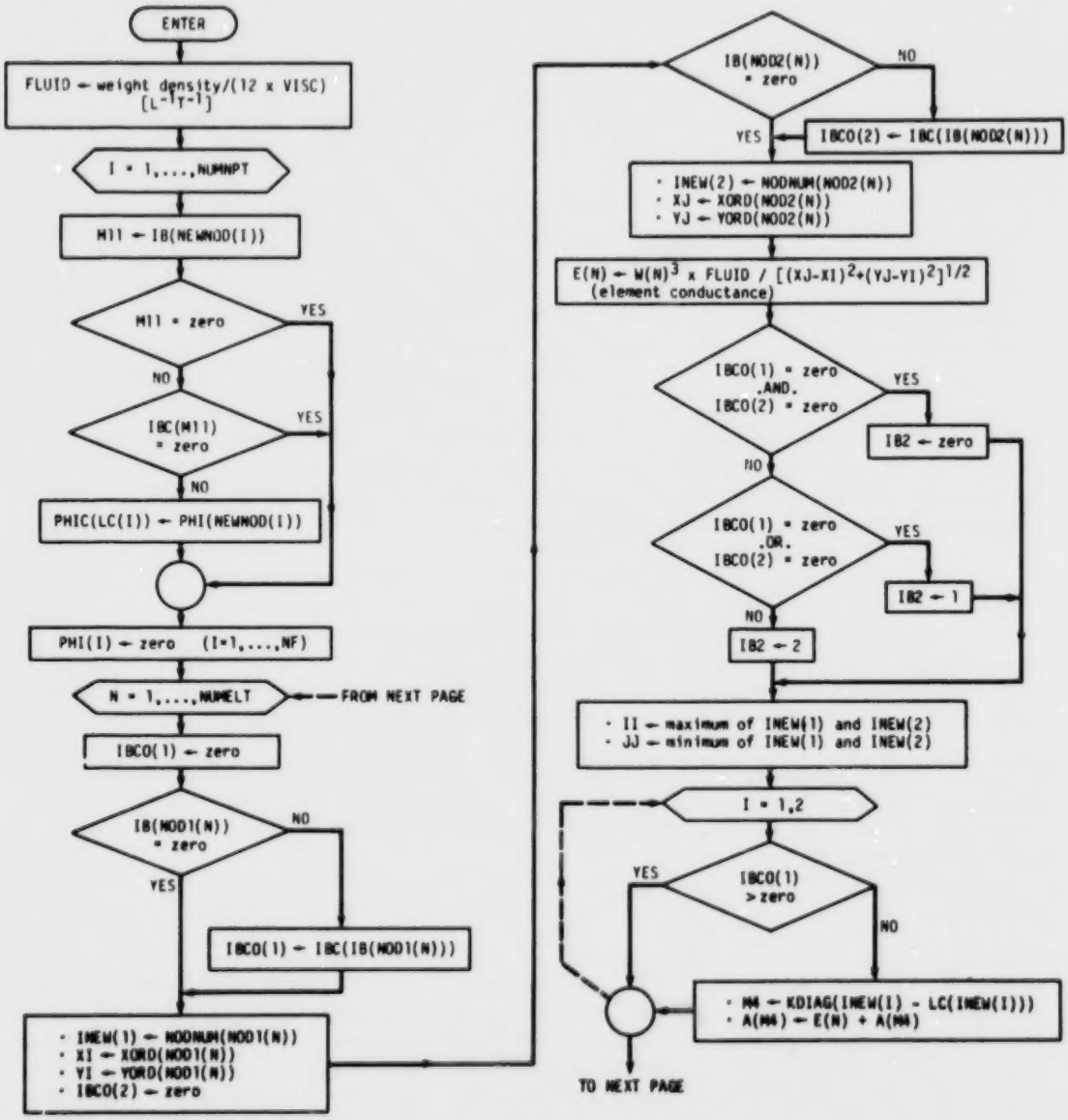


Figure 3.6. Flowchart of subroutine SOLPHI.

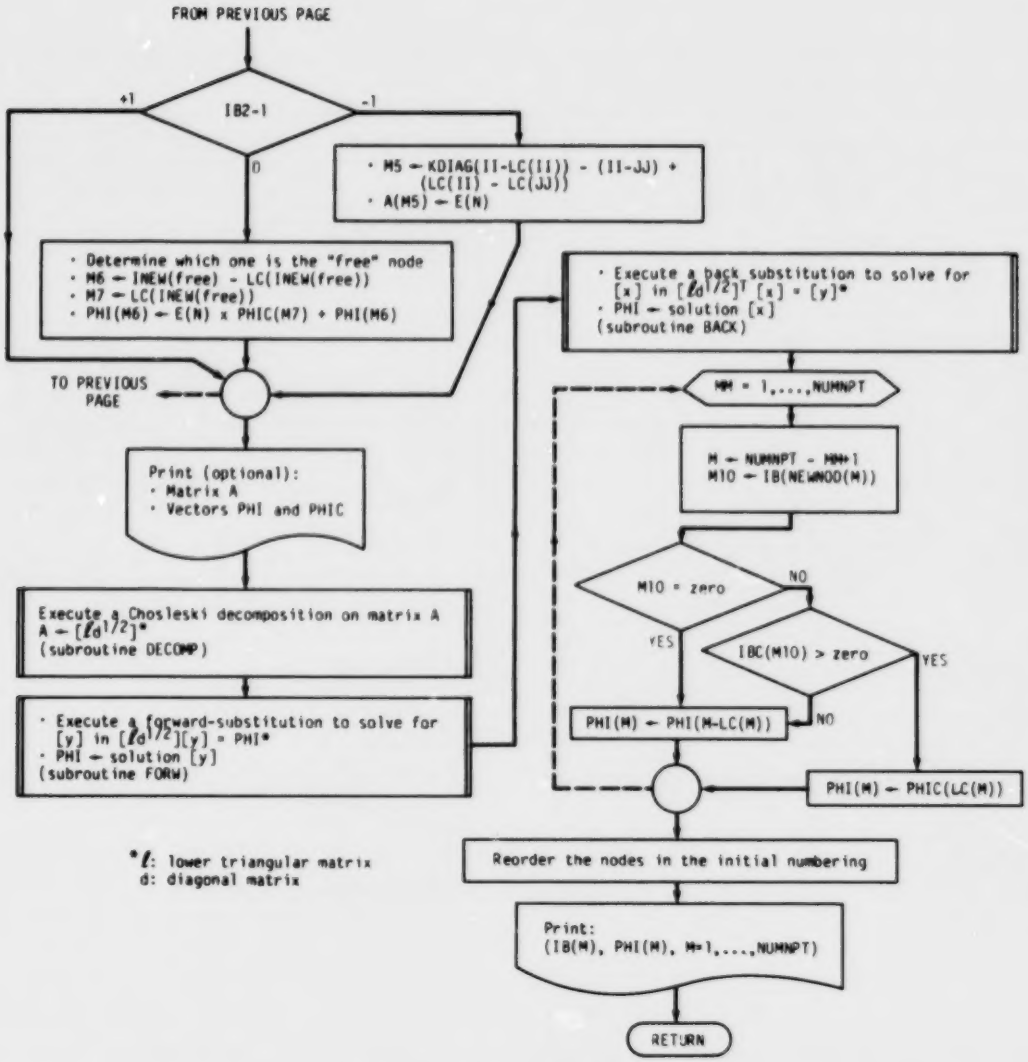


Figure 3.6. Continued.

After completing the head calculations, SOLPHI renumbers all the nodes in their initial numbering, so that the results can be more easily interpreted. In a final step before returning the execution to the main program, SOLPHI prints out the head value for each node.

At this stage, the subroutine FLOCAL (Fig. 3.7) takes over to compute the flow rate (FR), the flow velocity (VEL), and the Reynold number (RE) in every fracture segment using the following relationships:

$$FR_N = e_N \times (PHI_{node\ 1} - PHI_{node\ 2}) \qquad (3.14)$$

$$VEL_N = FR_N / W_N \qquad (3.15)$$

and

$$RE_N = |FR_N / \mu| \qquad (3.16)$$

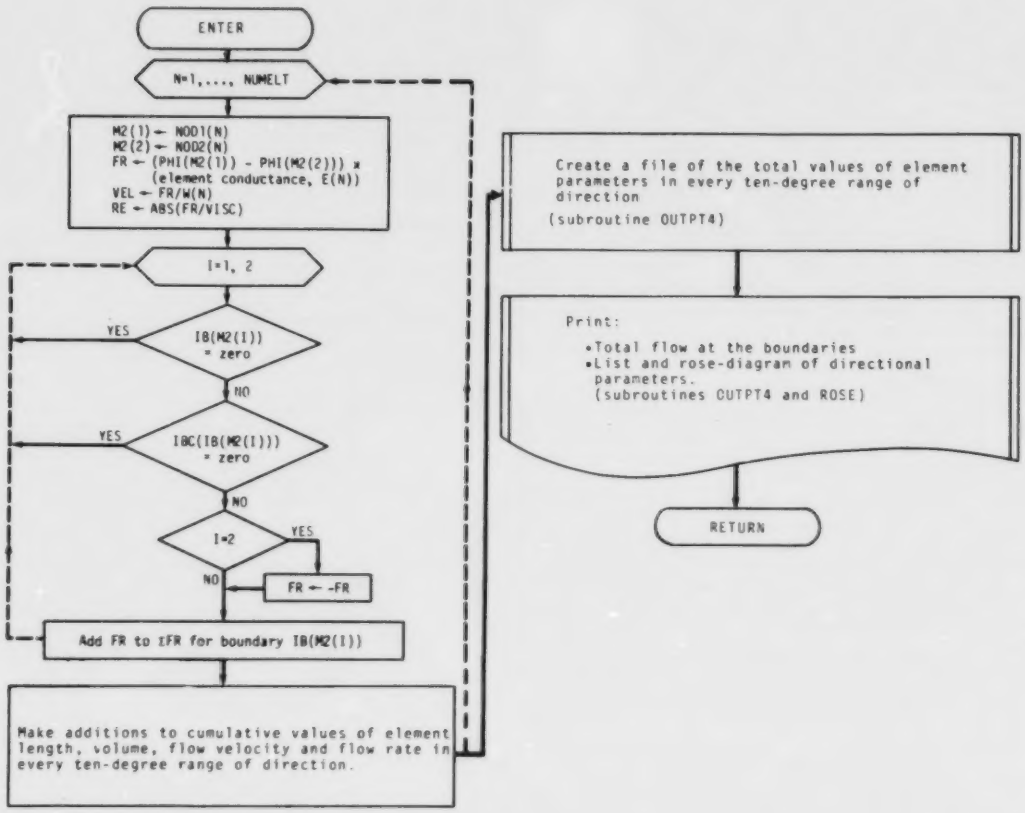


Figure 3.7. Flowchart of subroutine FLOCAL.

In the subprogram ELTDEF of NETWRK, the elements are defined in such a way that for any element the node number 1 has the lower y-value. In this convention, the flow rate calculated by (3.14) is positive in the direction of increasing y and negative otherwise, the direction with respect to x being immaterial. During these flow calculations, the cumulative flow rate is computed for all the elements with an extremity on a boundary. At the end of the flow calculations, the total flow rate is printed out for each boundary individually.

The subroutine FLOCAL also computes the total value of a number of variables in each ten-degree range of direction. The main directional variables computed in FLOCAL are the sum of (1) the element length l_N, (2) the surface area of fractures ($W_N \times l_N$), (3) the element permeabilities defined as $W_N^2 \times l_N$, and (4) the fluid velocities VEL_N and flow rates FR_N. In addition, the products of velocity and other parameters are also computed in every ten-degree range of direction. These products are used in the program NETRANS on an experimental basis to compute different weighted average velocities using different weighting factors (Sec. 4.2.2).

3.4 Program Verification and Accuracy Evaluation

In order to verify the results of the program NETFLO, a simple model has been used, constituted of two vertical fractures 10 m long that are linked together by a set of nine horizontal fractures 1 m apart (Fig. 3.8). Hydraulic heads of 0 and 100 m are imposed on the bottom and the top boundaries respectively. The theoretical flow rate in each one of the fractures can easily be computed using the cubic law for fluid flow between two smooth parallel plates (Equation 3.1). In these calculations, the values selected for fluid weight density (γ) and dynamic viscosity (μ) are 9.800×10^3 N/m^3 and 1.002×10^{-3} Pa.s respectively. Assuming an aperture of 10 μm for all of the fractures, the flow rate in each one of the vertical fractures is $-8.1503659 \times 10^{-9}$ m^3/s at eight digits accuracy. The minus sign means downward flow.

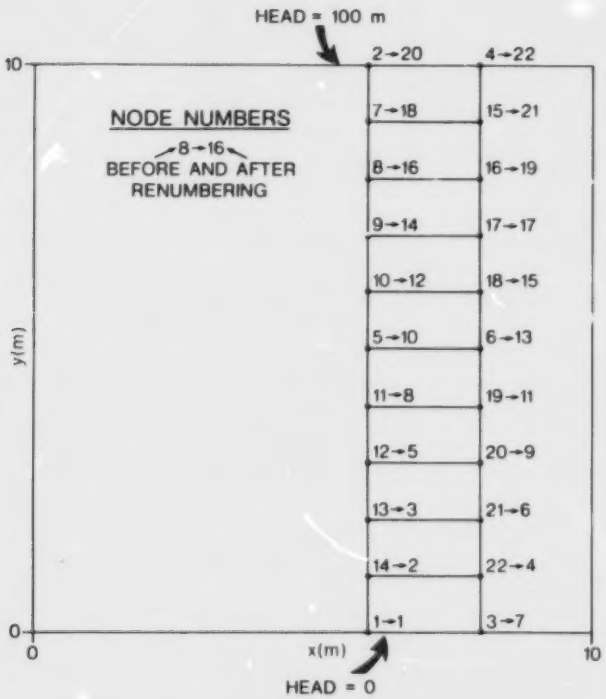


Figure 3.8. Fracture system used for evaluating the accuracy of the program NETFLO.

Before solving the fluid-flow problem depicted in Figure 3.8, the program NETFLO must first rearrange the node numbering in order to minimize the bandwidth of the node conductance matrix (Equation 3.5). The node numbers before and after renumbering are given in Figure 3.8. The initial numbering has purposely been made inefficient; it would yield a bandwidth of 19 (i.e., 22-3). Renumbering executed by the

subroutine OPTNUM, which is called by the subroutine OPSTOR, reduces the bandwidth to 4, resulting in a considerable economy in memory requirement.

A flow rate value has been computed by program NETFLO for each one of the fracture segments between two consecutive nodes in Figure 3.8. The computed values in the vertical segments vary between -8.1501810 and $-8.1506499 \times 10^{-9}$ m^3/s. Therefore, for this simple model, the maximum absolute error on flow rate is about 3×10^{-13} m^3/s, corresponding to a relative error of about 3.5×10^{-5}.

Since the aperture is uniform in each fracture and therefore the constitutive flow law (Equation 3.1) is linear, the numerical calculations should theoretically give the exact solution. In fact, the small numerical error that is observed is not a "discretization" error of the type most commonly encountered in numerical analysis. Instead, the observed error is the result of a number of round-off errors generated because the memories of all computing machines are finite and only a fixed number of digits (usually between 7 and 12) can be retained after each arithmetic operation.

Thus any irrational number and all numbers with more significant digits than can be retained that occur in a sequence of calculations must be approximated by rounded values. Detailed analysis of accumulated round-off error in the final results of numerical calculations is generally difficult because the magnitude and the sign of the error in each operation is unpredictable, particularly in a stochastically generated system. Even though probabilistic models can be applied to the propagation of rounding errors (Henrici, 1962), the simple empirical approach described in the following paragraphs suffices for the purpose of evaluating the accuracy of the results from the program NETFLO.

In order to evaluate the importance of the accumulated round-off error produced by program NETFLO for larger systems, the number of nodes (NN) between fixed-head boundaries in Figure 3.8 has been increased successively to 20, 50, and 100. The resulting errors in head and flow rate calculations are shown in Figure 3.9. For an unknown reason, the error in hydraulic head is always positive for NN = 10, 20, and 100, and always negative for NN = 50. It is also interesting to note that at each vertical distance, the error in one vertical fracture is, in general, slightly different from the error in the other vertical fracture; as a consequence there is a very small flow in each of the horizontal fractures, which should not be. These discrepancies, however, are relatively small and cannot be shown in Figure 3.9. The most important implication of Figure 3.9 is that, contrary to a discretization error, a round-off error is increased by increasing the number of nodes in a model.

Obviously, a model with a more complex geometry than the system of Figure 3.8 would produce error distributions that are less smooth than the curves of Figure 3.9. Even if it is strictly applicable only to the simple model of Figure 3.8, Figure 3.9

indicates, nevertheless, that the error in flow rate is maximal near a fixed-head boundary because of the steeper slope for the error in hydraulic head there. Moreover, the error in flow rate is of opposite sign at the inflow and at the outflow boundaries. Therefore a simple verification of the difference in flow rates or mass balance between

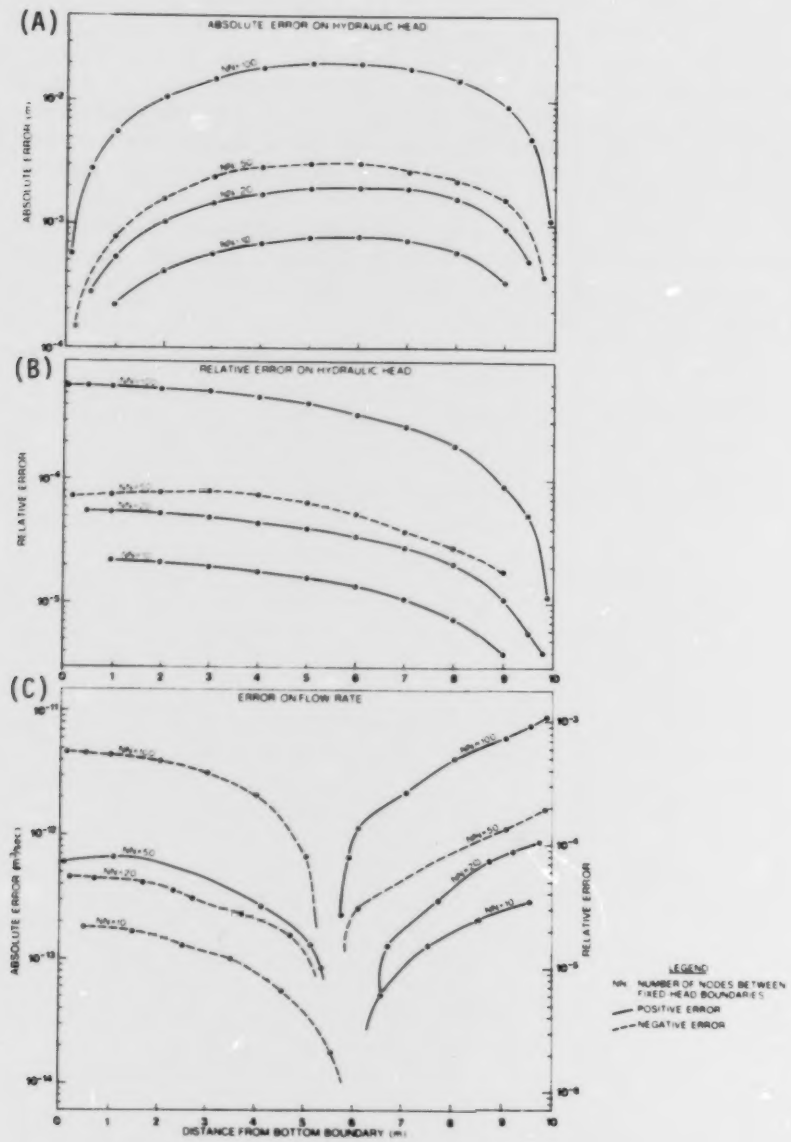


Figure 3.9. Distribution of accumulated round-off error for various number of nodes between fixed-head boundaries in the system of Figure 3.8: (A) absolute error on hydraulic head, (B) relative error on hydraulic head, (C) error on flow rate.

the inflow and the outflow can provide a conservative evaluation of the accuracy of the results for a particular model. Figure 3.10 indicates that a considerable increase in mass balance error results from an increase in the number of nodes. Figure 3.10 also shows that for a particular number of nodes in the system of Figure 3.8, a different fracture aperture yields a different relative mass balance error. However, this difference in relative error due to fracture aperture is relatively small and, moreover, it is not systematically in the same direction (i.e., a larger aperture can give either a larger or a smaller error). In a rough approximation, by extrapolating the trend of Figure 3.10, one can estimate that the relative mass balance error would be of the order of 10% for a flowpath containing between 200 and 500 nodes between the fixed-head boundaries.

It is worth pointing out that the cumulative round-off error depends both on the computer hardware and on the software. For

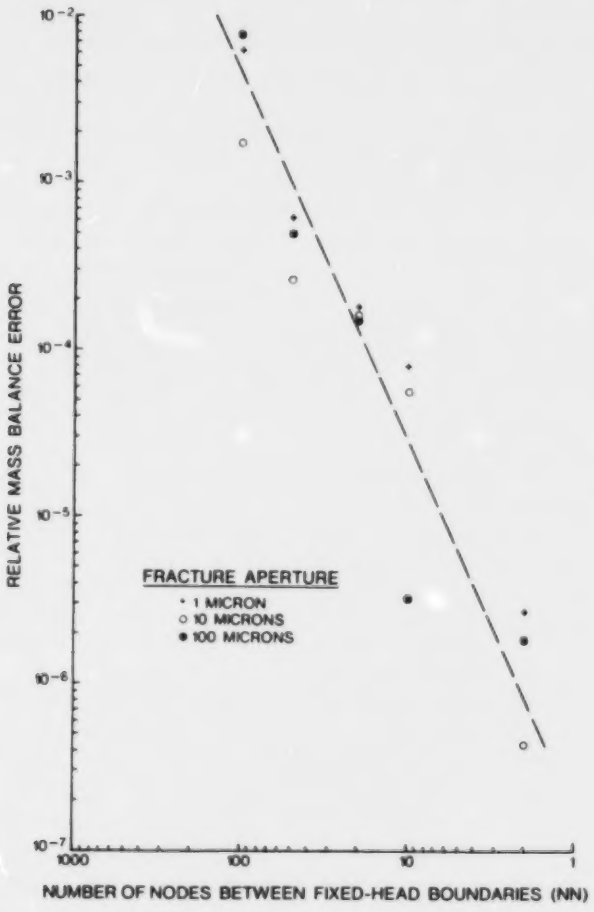


Figure 3.10. Relative error on mass balance as a function of number of nodes between fixed-head boundaries for the system of Figure 3.8.

instance, a different solution technique as well as a different computer system would produce cumulative errors that are different from the ones obtained in this study. Nevertheless, one can presume that the distribution of errors in a simple system like the one depicted in Figure 3.8 would be very similar to what is shown in Figure 3.9, even though the actual values would be different. No comparison has been made in this study between different solution techniques and different computer systems.

Since the round-off error is due to the limited size of memory reserved for a number after each arithmetic operation, an obvious corrective measure is to increase the size of the memory reserved for a few selected variables. The program NETFLO with DOUBLE PRECISION specified for the variables used in solving the fluid-flow problem produces a relative mass balance error of the order of 10^{-6} for all the model sizes represented in Figure 3.10. Considering this considerable improvement, DOUBLE PRECISION should be specified for selected arrays (particularly arrays A and PHI) when a computing machine with 32-bit words is used. Generally, 60-bit machines like the CYBER series do not require DOUBLE PRECISION variables.

3.5 Conclusion

The program NETFLO has been designed to be used jointly with the program NETWRK described in Chapter 2. The code is relatively fast and simple, and it can handle problems containing more than 3500 elements without exceeding 1 megabyte in computer storage even with DOUBLE PRECISION specified for selected variables. For these large networks, the computing time for nine complete flow calculations (corresponding to nine different aperture distributions) is in the order of 120 CPU seconds on an IBM 4341 machine. Therefore, the program NETFLO is considered as a reasonable compromise between code simplicity, on one side, and efficiency in computing time and memory storage, on the other side.

Stochastic Simulation of Particle Transport in a Fracture Network: Program NETRANS

4.1 Introduction

The movement of solute through fracture networks in rock has been investigated with physical models and with numerical models. Krizek *et al.* (1973) (see also Castillo, 1972) simulated the solute dispersion in a numerical model of continuous fractures, using an approximate concentration function at each node. This concentration function could assume either no-mixing or complete mixing at the nodes. Their assumptions were partly based on the results of physical experiments on node mixing and on dispersion in a single fracture (Krizek *et al.*, 1972; see also Socias *et al.*, 1979). Laboratory experiments on mixing at fracture intersections were also reported by Wilson and Witherspoon (1976).

Hull (1985) reported another series of physical experiments investigating the solute mixing at nodes and the dispersion in single fractures. He also described two types of physical models of fracture networks for dispersion studies: (1) rectangular networks of continuous openings in an impermeable matrix of plexiglass (see also Hull and Kuslow, 1982) and (2) a dual-porosity model of a rectangular network of discontinuous openings in a porous polyethylene material. As part of the same project, Miller (1983) developed a marker-particle model that simulates the solute dispersion in simple networks of discontinuous fractures and that accounts for flow in the matrix.

Schwartz *et al.* (1983) used stochastic tracking of particles to simulate solute dispersion in orthogonal networks of fractures of distributed size and semi-random location. A numerical transport model presented by Endo *et al.* (1984) considers the detailed movement of fluid within streamtubes through fracture networks. Finally, Robinson (1984) developed computing algorithms for both a mass-lumping approach and a particle-following approach.

The numerical model NETRANS presented here is different from the models mentioned above. It uses a second-level stochastic process based on statistics of directional parameters computed in the program NETFLO (Chap. 3) to follow particles in a virtual network of arbitrary size (Rouleau, 1987). This latter approach has some similarities with the approximation of the unit hydrograph of a surface drainage basin by assuming flow through random channel networks (e.g., Troutman and Karlinger, 1985).

4.2 Program NETRANS

Program NETRANS (Figs. 4.1 and 4.2) uses the average value computed for each ten-degree range of direction for the three parameters: flow rate, flow velocity, and segment (or element) length. NETRANS follows a number of particles, one at a time, through a virtual fracture network. At each step (or virtual node) in the tracking of a particle, the flow direction is determined stochastically by the relative importance of the average flow rate in each ten-degree range of direction. After the flow direction is selected, the particle is moved a distance equal to the average segment length in that direction. The particle velocity is equal to a weighted average velocity (see Sec. 4.2.2), also in that direction. At the end of each step, the new coordinates of the particles, as well as the incremented time, are recorded. The tracking of a particle is stopped after a specified distance has been reached.

In the present version of NETRANS, the distance a particle moves in a given step is simply the mean of the length of all the elements with the flow in the corresponding direction. Further developments of the program could include the stochastic determination of the displacement distance in a manner similar to the flow direction. The determination of the flow direction itself and of the flow velocity merits further explanations.

4.2.1 Determining the Flow Direction

At the end of execution of the program NETFLO (Sec. 3.3.2), one of the directional parameters recorded is the cumulative value of flow rate over all the elements in every ten-degree range of direction (DFR). The program NETRANS computes the ratio of each one of these cumulative directional flow rates over the sum of the flow rates in all

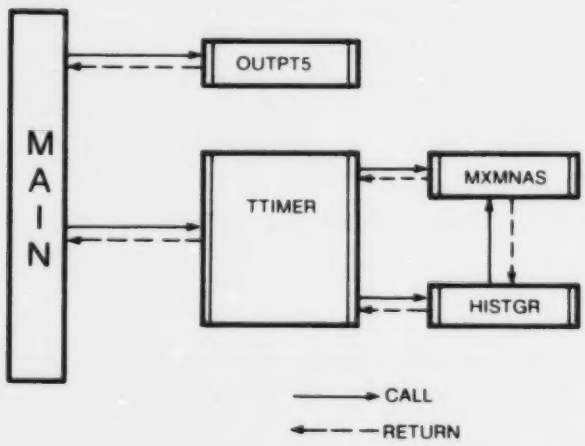


Figure 4.1. Structure of program NETRANS.

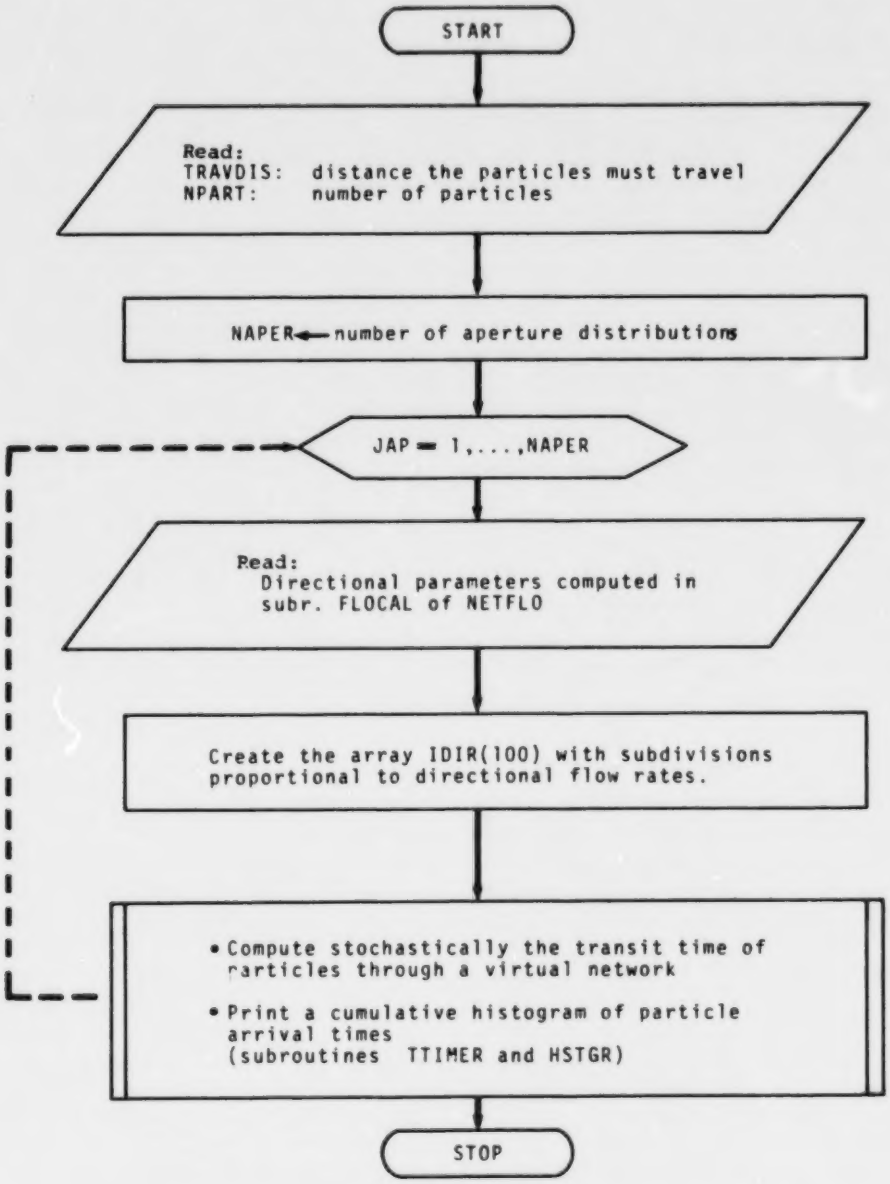


Figure 4.2. Flowchart of the main program in NETRANS.

directions. The resulting ratio values (RDFR) are then used to subdivide the range [0,100]: each subdivision is given a value of a direction index IDIR (between 1 and 36); the size of each subdivision is proportional to the value of RDFR for the corresponding direction range (Fig. 4.3). Subsequently, the subroutine TTIMER (Fig. 4.4) uses the array IDIR, jointly with random numbers uniformly distributed in the range [0,100], to determine the direction of flow at each step of the particle tracking.

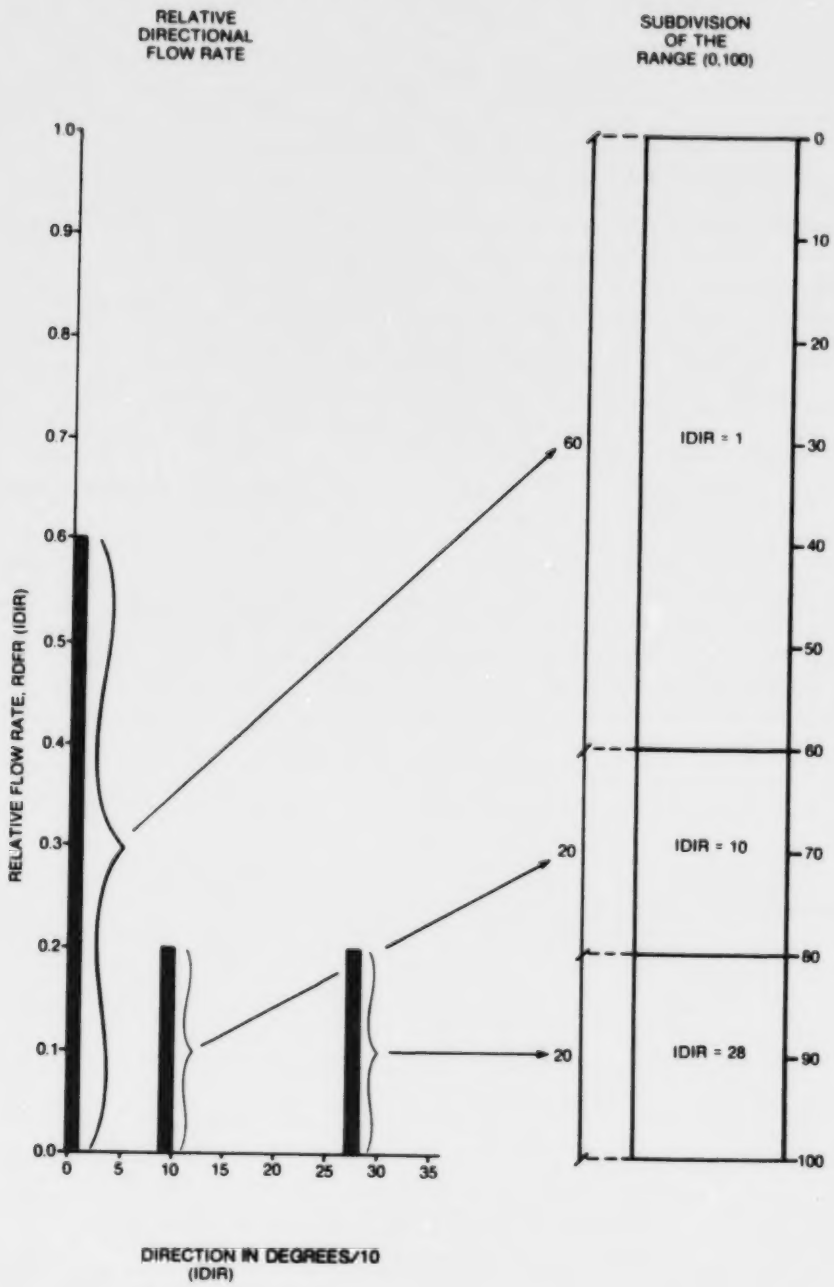


Figure 4.3. Set-up of the array IDIR (100) for the stochastic determination of the flow direction.

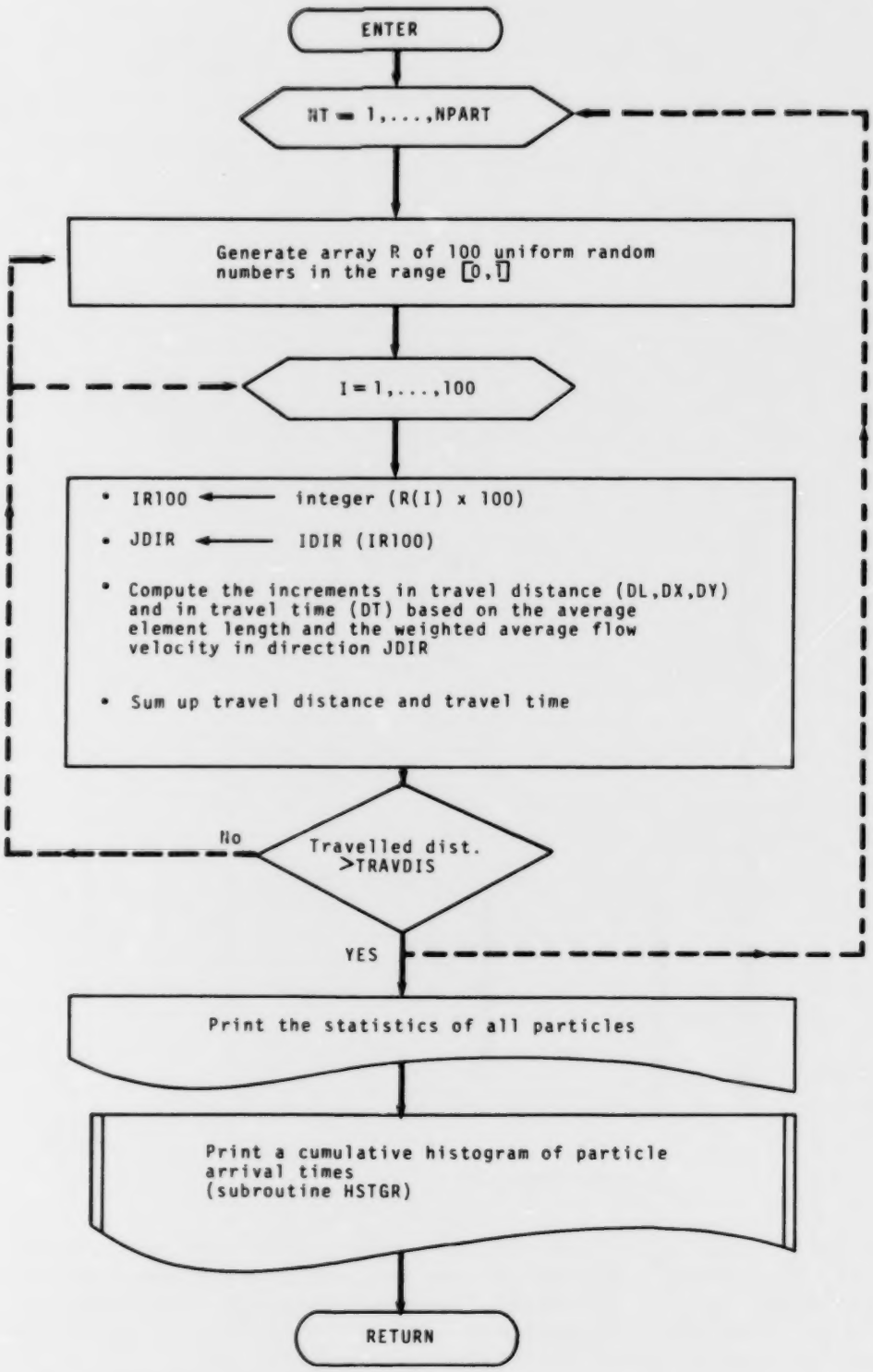

ENTER
MT = 1,...,NPART
Generate array R of 100 uniform random numbers in the range [0,1]
I = 1,...,100
• IR100 ← integer (R(I) x 100)
• JDIR ← IDIR (IR100)
• Compute the increments in travel distance (DL,DX,DY) and in travel time (DT) based on the average element length and the weighted average flow velocity in direction JDIR
• Sum up travel distance and travel time
No
Travelled dist. >TRAVDIS
YES
Print the statistics of all particles
Print a cumulative histogram of particle arrival times (subroutine HSTGR)
RETURN

Figure 4.4. Flowchart of subroutine TTIMER.

4.2.2 *Determining the Average Flow Velocity in Every Direction*

The simple average directional velocity is not necessarily the velocity the solute particle moves on average in a given direction. Presumably this latter average particle velocity is also affected by the flow rate in the elements and by the element lengths. For that reason I have tried a number of different weighting factors to compute weighted average directional velocities. Comparisons of the resulting distributions of transit times of stochastic particles through the sample network of Figure 4.5, with the deterministic results of transit time assuming plug flow, are given in Figure 4.6 for four different weighting factors for the average directional velocity. A weighting factor of one is applied by calculating the cumulative value of the simple flow velocity for all the elements in a direction range, and dividing that cumulative velocity by the number of elements in that direction range. The three other weighting factors considered are the element length (l), the flow velocity itself (v), and the product of the element length and the flow velocity in that element (l x v).

By comparing the 50% mass arrival time on the curves for the four weighting factors considered (Fig. 4.6), it is apparent that the factor that gives the most accurate results is the product (l x v). Clearly, everything else being equal, a higher flow velocity in a fracture segment means a proportionally higher flow rate, and therefore a higher probability that a particle is travelling at that velocity. Also, the longer a fracture segment, the more influence the flow parameters of this segment have on the overall particle displacements.

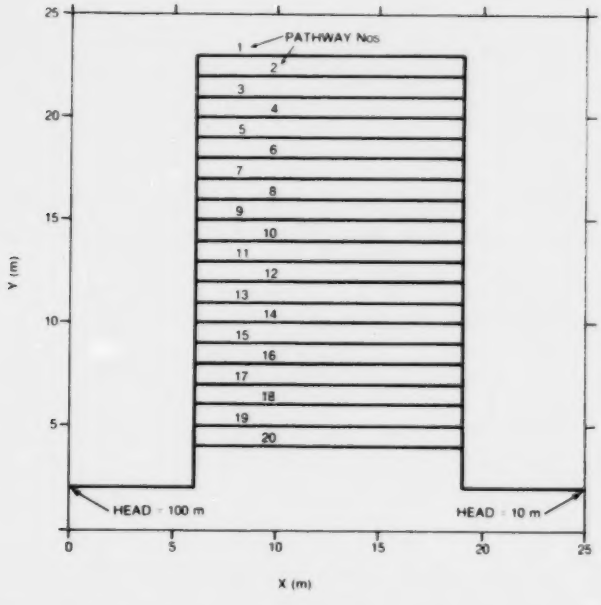


Figure 4.5. Fracture system used for NETRANS verification. (The aperture of all the fractures is 5 μm.)

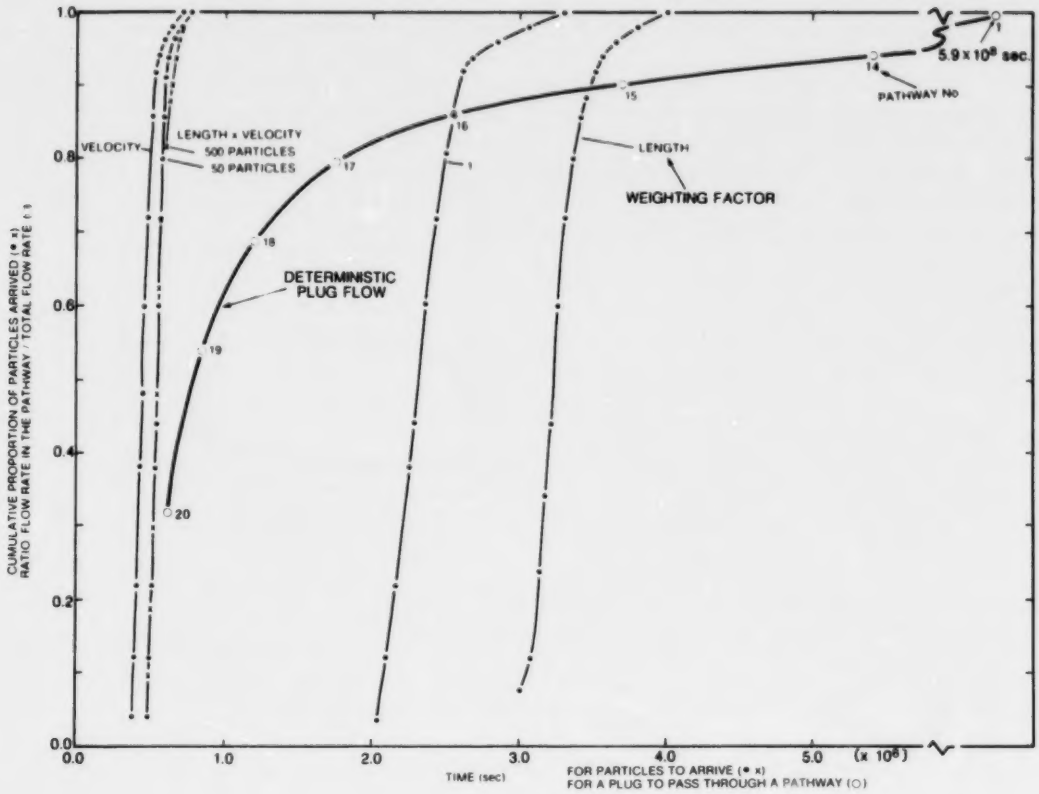


Figure 4.6. Comparison of deterministic plug flow and the stochastic transit time of particles with different weighting factors for the flow velocity.

A more comprehensive analysis of possible weighting factors should include the element aperture and flow rate, and their product with the element length.

4.2.3 Number of Particles Required

Figure 4.6 also shows the results of a trial run with 500 particles instead of 50. The curve for 500 particles is not significantly different from the curve for 50, indicating that 50 particles are enough to obtain a good estimate of transit times.

4.3 Conclusion

In its present version, the program NETRANS can be used to estimate the average transit time of particles over any distance in a fracture network. The distance travelled by the particles can be much larger than the size of the original fracture network generated by the program NETWRK and considered by the fluid-flow program NETFLO.

Obviously a number of simplifying assumptions are implicitly included in the program NETRANS, and these limitations must be considered when interpreting the results. For instance, we assume an impermeable matrix and we neglect dispersion within each fracture. Also Figure 4.6 shows that the shape of the particle breakthrough curves is significantly different (much less dispersion) from the shape of the deterministic plug-flow breakthrough. This lack-of-fit indicates that the present version of NETRANS cannot approximate the shape of solute breakthrough curves in fracture networks. Modifications to the program NETRANS that could improve the realism of the simulated breakthrough curves include the replacement of the average value by distributed values for the segment length and for the weighted flow velocity in every direction range.

The program NETRANS in its present version can be applied only to rectangular models. Attempts to develop other subroutines to be used with circular models were faced with the problem of particles that never reach the inner circular boundary in the case of converging flow. Indeed, for many particles, the virtual pathway went beside the inner boundary and missed it completely. The case of diverging flow is easier to handle, and this could be done with minor modifications to the program; for instance, by expressing the travelled distance in terms of radial coordinates.

General Conclusion

The stochastic discrete fracture (SDF) model presented here is a practical tool to incorporate the statistical geometry of a fracture system in the investigation of ground-water flow and mass transport in fractured rock masses. The fracture-geometry parameters of interest are the fracture densities and the distributions of fracture orientations, sizes, and apertures.

The computer programs NETWRK, APEGEN, NETFLO, and NETRANS have been developed using a modular approach. Their execution sequence must correspond to the order of presentation in this report. Each program, however, is executed individually, and its results can be consulted before executing another program. This modular structure gives more flexibility to the package. Also, the programs were designed to provide a reasonable compromise between memory storage, computing time, and code complexity. Those factors make this SDF package suitable for site-specific simulations.

References

Andersson, J., A.M. Shapiro, and J. Bear. 1984. A stochastic model of a fractured rock conditioned by measured information. Water Resour. Res., 20: 79-88.

Asfari, A., and P. A. Witherspoon. 1973. Numerical simulation of naturally fractured reservoirs. Soc. Pet. Eng. of AIME, Paper SPE-4290, 8 pp. Dallas, Texas.

Baczynski, N.R.P. 1980. Zonal concept for spatial distribution of fractures in rock. In Proc. 3rd Australia-New Zealand Conf. Geomech., vol. 2, 29-33. Wellington, New Zealand.

Baecher, G. B. 1983. Statistical analysis of rock mass fracturing, Math. Geol., 15: 329-48.

Bridges, M. C. 1975. Presentation of fracture data for rock mechanics. In Proc. 2nd Australia-New Zealand Conf. Geomech., 144-48. Brisbane, Australia.

Caldwell, J. A. 1972. The theoretical determination of the permeability tensor for jointed rocks. In Percolation through fissured rock, Proc. Symp. Int. Soc. Rock Mech. and Int. Assoc. Eng. Geol., T1C1-6. Stuttgart.

Castillo, R.E. 1972. Dispersion of contaminant in jointed rock. Ph.D. thesis, 192 pp. Northwestern University, Evanston, Ill.

Castillo, E., G. M. Karadi, and R. J. Krizek. 1972. Unconfined flow through jointed rock. Water Resour. Bull., 8(2): 266-81.

Collins, R.J. 1973. Bandwidth reduction by automatic renumbering. Int. J. Num. Methods Eng., 6: 345-56.

Conrad, F., and C. Jacquin. 1973. Représensation d'un réseau bi-dimensional de fractures par un modèle probabiliste--application au calcul des grandeurs géométriques des blocs matriciels. Rev. Inst. Fr. Pet., 28(6): 843-90.

Dershowitz, W. S., B.M. Gordon, and J. C. Kafritsas. 1985. A new three-dimensional model for flow in fractured rock. In Proc. Int. Congress on Hydrogeology of Rocks of Low Permeability, 441-48. Int. Assoc. of Hydrogeologists, Tucson.

Duguid, J.O. and P.C.Y. Lee. 1977. Flow in fractured porous media. Water Resour. Res., 13: 558-66.

Endo, H.K., J.C.S. Long, C.R. Wilson, and P. A. Witherspoon. 1984. A model for investigating mechanical transport in fracture networks. Water Resour. Res., 20(10): 1390-1400.

Fawcett, R.J., S. Hibbert, and R.N. Singh. 1984. An appraisal of mathematical models to predict water inflows into underground coal workings. Int. J. of Mine Water, 3: 33-54.

Gale, J.E. 1977. A numerical, field and laboratory study of flow in rocks with deformable fractures. Sci. Ser. No. 72, 145 pp. Water Resources Branch, Inland Waters Directorate, Environment Canada, Ottawa, (Also, Ph.D. diss., University of California, Berkeley, 1975.)

Gale, J.E., and A. Rouleau. 1984. Characterizing and interpreting the geometry, permeability and porosity of fractures for repository evaluation. In Proc. Int. Symp. Field Measurements in Geomech., Zurich, ed. K. Kovari, 1343-69. Rotterdam: A.A. Balkema.

Henrici, P. 1962. *Discrete Variable Methods in Ordinary Differential Equations*, 407 pp. New York: John Wiley.

Hudson, J.A., and P.R. LaPointe. 1980. Printed circuits for studying rock mass permeability. Technical Note, Int. J. Rock Mech. Min. Sci., 17: 297-301.

Hudson, J.A., and S.D. Priest. 1979. Discontinuities and rock mass geometry. Int. J. Rock Mech. Min. Sci., 16: 239-362.

Huitt, J.L. 1956. Fluid flow in simulated fractures. AIChE J. 2(2): 259-64.

Hull, L.C. 1985. Physical model studies of despersion in fracture systems. Idaho Nat. Eng. Lab. Informal Report, EGG-ELS-6845, 74 pp. Idaho Falls, Idaho.

Hull, L.C., and K.N. Koslow. 1982. Dispersion in fracture networks. In Proc. 8th Stanford Geothermal Reservoir Eng. Conf., 258-88, Stanford University, Stanford, Calif.

Huskey, W.L., and P.B. Crawford. 1967. Performance of petroleum reservoirs containing vertical fractures in the matrix. Soc. Pet. Eng. J., June 1967: 221-28.

Isaacs, L.T., and K.G. Mills. 1980. Linear theory methods for pipe networks analysis. J. Hydraul. Div. Proc. Am. Soc. Civ. Eng., 106(HY7): 1191-1201.

Jennings, A. 1977. *Matrix Computation for Engineers and Scientists*, 330 pp. New York: John Wiley.

Kesavan, H.K., and M. Chandrashekar. 1972. Graph-theoritic model for pipe network analysis. J. Hydraul. Div. Proc. Am. Soc. Civ. Eng., 98(HY2): 345-64.

Krizek, R.J., E. Castillo, and G.M. Karadi. 1973. Theoretical study of dispersion in a fractured rock aquifer. J. of Geophys. Res., 78(3): 558-73.

Krizek, R.J., G.M. Karadi, and E. Socias. 1972. Dispersion of a contaminant in fissured rock. In Percolation through fissured rock, Proc. Symp. Int. Soc. Rock Mech. and Int. Assoc. Eng. Geol., T3C1-15. Stuttgart.

LaPointe, P.R., and J.A. Hudson. 1985. Characterization and interpretation of rock mass jointing patterns. Geol. Soc. Am., Spec. Pap., 199, 37 pp. Boulder, Colorado.

Long, J.C.S., P. Gilmour, and P.A. Witherspoon. 1985. A model for steady fluid flow in random three-dimensional networks of disc-shaped fractures. Water Resour. Res., 21: 1105-15.

Long, J.C.S., J.S. Remer, C.R. Wilson, and P.A. Witherspoon. 1982. Porous media equivalents for networks of discontinuous fractures. Water Resour. Res., 18(3): 645-58.

Louis, C. 1969. A study of groundwater flow in a jointed rock and its influence on the stability of rock masses. Imperial College Rock Mech. Res. Rep. No. 10, 90 pp. London, U.K.

Marcus, H., and D.E. Evenson. 1961. Directional permeability in anisotropic porous media. University of California, Berkely, Water Resources Center Contrib. 31, 105 pp.

Mah, R.S.H., and M. Shacham. 1978. Pipe network design and synthesis. In *Advances in Chemical Engineering*, ed. T.B. Drew *et al.*, 125-209. New York: Academic Press.

Miller, J.D. 1983. A fundamental approach to the simulation of flow and dispersion in fractured media. In Proc. 9th Stanford Geothermal Reservoir Eng. Conf., 373-79. Stanford University, Stanford, Calif.

Noorishad, J., M.S. Ayatollahi, and P.A. Witherspoon. 1982. A finite-element method for coupled stress and fluid flow analysis in fractured rock masses. Int. J. Rock Mech. Min. Sci., 19: 185-193.

Ollos, G. 1963. Examen hydraulique de l'écoulement dans des roches crevassées sur des modèles réduits. Bull. Int. Assoc. Sci. Hydrol., 8(2): 9-22.

Parsons, R.W. 1966. Permeability of idealized fractured rock. Soc. Pet. Eng. J., June 1966: 126-36.

Prats, M. 1972. The influence of oriented arrays of thin impermeable shale lenses or of highly conductive natural fractures on apparent permeability anisotropy. J. Pet. Technol., Oct. 1972: 1219-21.

Robinson, P.C. 1982a. Connectivity of fracture systems: a percolation theory approach. Theor. Phys. Div., At. Energy Res. Establ., Rep. TP 918, 35 pp. Harwell, U.K.

Robinson, P.C. 1982b. NAMNET: network flow program. At. Energy Res. Establ., Rep. R-10510, 28 pp. Harwell, U.K.

Robinson, P.C. 1984. Connectivity, flow and transport in network models of fractured media. Ph.D. thesis, Oxford University. (Also At. Energy Res. Establ., Rep. TP 1072, Harwell, U.K.)

Rouleau, A. 1987. A stochastic particle transport model based on directional statistics of flow through fracture networks. Actes du NATO Advanced Research Workshop on Advances in Analytical and Numerical Groundwater Flow and Quality Modelling. Lisboa: D. Reidel Publ.

Rouleau, A., and J.E. Gale. 1985. Statistical characterization of the fracture system in the Stripa granite, Sweden. Int. J. Rock Mech. Min. Sci., 22: 353-67.

Rouleau, A., and J.E. Gale. 1987. Stochastic discrete fracture simulation of groundwater flow into an underground excavation in granite. Int. J. Rock Mech. Min. Sci., 24: 99-112

Samaniego, A., and S.D. Priest. 1984. The prediction of water flows through discontinuity networks into underground excavations. In Proc. Symp. Design and Performance of Underground Excavations, E.T. Brown and J.A. Hudson, 157-64. Int. Soc. for Roch Mech., Cambridge.

Schwartz, F.W., L. Smith, and A.S. Crowe. 1983. A stochastic analysis of macroscopic dispersion in fractured media. Water Resour. Res., 19(5): 1253-65.

Shapiro, A.M., and J. Anderson. 1985. Simulation of steady-state flow in three-dimensional fracture networks using the boundary-element method. Adv. Water Resour. 8: 106-110.

Smith, L., and F.W. Schwartz. 1984. An analysis of the influence of fracture geometry on mass transport in fractured media. Water Resour. Res., 20: 1241-52.

Snow, D.T. 1969. Anisotropic permeability of fractured media. Water Resour. Res., 5: 1273-89.

Socias, E.J., R.J. Krizek, and R.H. Borden. 1979. Model study of hydrodynamic dispersion in jointed rock. In Proc. World Congress on Water Resources, 2967-73. Mexico City, Mexico.

Troutman, B.M., and M.R. Karlinger. 1985. Unit hydrograph approximations assuming linear flow through topologically random channel networks. Water Resour. Res., 21(5): 743-54.

Veneziano, D. 1978. Probabilistic model of joints in rock. Res. Rep. Dept. Civ. Eng., 46 pp. Massachusetts Institute of Technology, Cambridge, Mass.

Wilson, C.R., and P.A. Witherspoon. 1970. An investigation of laminar flow in fractured porous rocks. Internal rep., Dept. Civ. Eng., 178 pp. University of California, Berkeley.

Wilson, C.R., and P.A. Witherspoon. 1976. Flow interference effects at fracture intersections. Water Resour. Res., 12(1): 102-104.

Wittke, W. 1970. Three-dimensional percolation of fissured rock. In Planning open pit mines, Proc. Symp. Inst. Mine Met., ed. P.W.J. van Rensburg, 181-91. Johannesburg, South Africa.

Appendix A

List of the Main Variables Used in the Programs NETWRK (NW), APEGEN (AP), NETFLO (NF), and NETRANS (NT)

List of the Main Variables Used in the Programs NETWRK (NW), APEGEN (AP), NETFLO (NF), and NETRANS (NT)

A.1 List of Input Parameters

AAP(I) and BAP(I) (NW,AP): first and second parameters of the distribution of apertures for fracture set I. For APDSTR=1, AAP is the aperture value and BAP is not used. For APDSTR=2, AAP and BAP are respectively the mean and the standard deviation of the (natural) log-transformed aperture. Note that the aperture is expressed in metres; (m).

ALE(I) and BLE(I) (NW): first and second parameters of the distribution of trace lengths for the fracture set I. For LDSTR=1, ALE is the trace length and BLE is not used. For LDSTR=2, ALE and BLE are respectively the mean and the standard deviation of the (natural) log-transformed trace length values; (m).

APDSTR(I) (NW, AP): type of distribution for apertures of fracture set I: (1) single-valued variable, (2) log-normal distribution; see AAP and BAP.

ATH(I) and BTH(I) (NW): first and second parameters of the distribution of trace angles within the simulation plane for fracture set I. For THDSTR=1, ATH is the angle value and BTH is not used. For THDSTR=2, ATH and BTH are respectively the mean and the standard deviation of the angle values. ATH(I) should be within the range 0°-180°. Angles are expressed counter-clockwise from the positive x-axis; see Figure D.1; (degree).

ATHP(I) and BTHP(I) (NW): first and second parameters of the distribution of angles between the simulation plane and the fractures of set I; ATHP(I) should be within the range 0°-180°. Angles are expressed as

	measured within the simulated rock slice, on the right-hand side of the fracture trace while looking in the direction ATH(I); see Figure D.1; (degree). For THDSTR=1, ATHP is the angle value and BTHP is not used. For THDSTR=2, ATHP and BTHP are respectively the mean and the standard deviation of the angle values.
BAP(I) (NW, AP):	see AAP(I).
BLE(I) (NW):	see ALE(I).
BTH(I) (NW):	see ATH(I).
BTHP(I) (NW):	see ATHP(I).
COEF (NW):	coefficient used to multiply a preliminary estimated number of fractures in a set, in order to get a value of ESTNFR (see subroutine GENLIN) large enough for most realizations; typically COEF should be in the range 2.0 to 6.0.
DAAP (NW,AP):	increment between different values of the aperture distribution parameter AAP.
DENS(I) (NW):	density of fracture set I, defined as the surface area of fractures per unit volume of rock; (m^{-1}).
HDB(I) and HDE(I) (NF):	input value of hydraulic head at the beginning and at the end point respectively, of the boundary segment I; moving clockwise along the boundary; not used for the no-flow boundaries; see Figure D.3; (m).
IBC(I) (NW,NF):	code for the boundary condition for flow along the boundary segment I; (0) no-flow bndr., (1) constant head along the bndr., (2) linearly decreasing head along the bndr., (3) logarithmically decreasing head along the bndr.; in the latter case, the bndr. must not cross a zero axis; see Figure D.3.
IGEOM (NW,NF,NT):	code for the general geometry of the model; (1) for a rectangular model, (2) for a circular model; see Figure D.3.
IOR (AP):	code for the aperture generation mode; (0) if no use is to be made of orientation data, (1) if the orientation

	is to be considered; the second option is not operational yet.
IPRT (NW,AP,NF,NT):	code for the level of information desired in the output files; possible values are 1, 2, and 3, with increasing level of detail (see Table B.2). Also, the fracture and the node numbers are not written on the computer plot with IPRT = 1.
ISHAP(I) (NW,NF):	code for the shape of the boundary segment I; (0) when its length is zero, (1) for a linear segment, (2) for a circular segment; see Figure D.3.
ISP (NW):	code for the type of sampling for spacing values between traces of fractures of the same set; (0) if no sampling is desired, (1) for sampling along lines perpendicular to the mean orientation of the traces (subroutine SPCNG1), (2) for sampling along randomly oriented lines (subroutine SPCNG2).
ITERM (I) (NW):	termination mode for fracture set I; (0) free extremities, (1) extremity one (i.e., higher y-value) is abutting on fractures of any set with index <I; ITERM(1) cannot equal 1.
LDSTR(I) (NW):	type of distribution of trace lengths for the fracture set I; (1) for a single-valued variable, (2) for a log-normal distribution; see ALE and BLE.
MPART(10) (NW):	array of particle numbers for which a detailed record of the track is desired; there are NMAP such particles.
NAAP (NW,AP,NF,NT):	number of different values of AAP to be used in the aperture generation.
NAP (NW,AP,NF,NT):	number of aperture distributions to be generated with the same distribution parameters.
NMAP (NT):	number of particles for which a detailed record of the track is desired (see MPART).
NPART (NT):	number of particles to be tracked (generally 50).

NSET (NW,AP,NF):	number of fracture sets; maximum = 5.
PLTSIZ (NW):	length (in inches) of the longer axis of the plot to be generated; if PLTSIZ $\leq$ 0, no plot is made.
R(I) (NW):	radius of the boundary segment I; it is used only for circular segment, i.e., with ISHAP(I)=2; see Figure D.3; (m).
SEED (NW,AP):	seed number for the random number generation routines; double precision number in the range 1 to 2147483647.
THDSTR(I) (NW):	type of distribution of angles for the fracture set I; (1) for a single-valued variable, (2) for a normal distribution; see ATH, BTH, ATHP, and BTHP.
TRAVDIS (NT):	distance a particle must travel to make it through the system (m); its value does not have to be the same as the size of the model for the generated network.
XB(I) and YB(I) (NW,NF):	coordinates of the beginning of the boundary segment I, moving clockwise along the boundaries; (m).
XMAXI, XMINI, YMAXI YMINI (NW):	inner boundaries of the rectangular model; see Figure D.3; (m).
XMAXO, XMINO, YMAXO and YMINO (NW):	outer boundaries of the rectangular model; required even for a circular geometry model; see Figure D.3; (m).
YB(I) (NW,NF):	see XB(I).
YMAXI and YMINI (NW):	see XMAXI.
YMAXO and YMINO (NW):	see XMAXO.

A.2. List of Parameters Used Internally

A(MAXA) (NF):	submatrix of the node conductance matrix; it contains information only on elements that are free at both ends; A is symmetric and uses a variable-bandwidth storage scheme (see KDIAG).
ACTLEN (NW):	actual total length of the set being generated at the end of the generation process; (m).

AP(J) (NW):	aperture of fracture J in the set being processed; (m).
CUMX and CUMY (NT):	cumulative travelled distance in x- or y-direction respectively for the particle being traced; (m).
CUMT (NT):	cumulative travel time for the particle being traced; (s).
CUT1(J,I) and CUT2(J,I) (NW):	record of whether or not extremity 1 and 2 respectively of line (J,I) intersects a boundary (bndr.) and a bndr. of what type: (0) does not cut a bndr., (1) cuts a no-flow bndr., (2) cuts a fixed-head bndr.
DEADLE(I) (NW):	total length of dead segments in fracture set I; (m).
DEADVO(I) (NW):	total "volume" (i.e., length x aperture x 1) of dead segments in fracture set I; (m^3).
DEGRE(J) (NW):	trace angle within the simulation plane of fracture J in the set being processed; same convention as for ATH(I); (degree).
DEGREP(J) (NW):	angle between the simulation plane and fracture J in the set being processed; same convention as for ATHP(I); (degree).
DESLEN (NW):	desired total length of fracture traces for the set being processed; (m).
DFR(K) (NF, NT):	sum of flow rates in the elements within the range of direction K, with K=1 to 36; (m^3/s)
DK(K) (NF):	sum of (non-directed) element "permeabilities" in the direction K, with K=1 to 18; the permeability of an element is defined as the aperture squared, weighted by the element length, i.e., W^2 X ELLEN; (m^3).
DLE(K) (NF, NT):	sum of (directed) element lengths (ELLEN) in the range of direction K, with K=1 to 36; (m).
DLE180(K) (NF):	sum of (non-directed) element lengths (ELLEN) in the range of direction K, with K=1 to 18; (m).

DLM(K) (NT): mean length of (directed) elements in the range of direction K, with K=1 to 36; (m).

DLVEL(K) (NF,NT): sum of the products ELLEN x VEL for all the elements in the range of direction K, with K=1 to 36; (m^2/s). DLVEL is used as the sum of the weighting factors in the computation of the weighted average directional velocity DWTVM(K).

DLVEL2(K) (NF,NT): sum of the weighted flow velocities in the elements within the range of direction K, with K=1 to 36. The weighted flow velocity in an element is defined as the product (ELLEN x VEL) x VEL; (m^3/s^2).

DPOR(K) (NF): sum of (non-directed) element "porosities" in the range of direction K, with K=1 to 18; the porosity of an element is defined as the product ELLEN x W; (m^2).

DVEL(K) (NF,NT): sum of the flow velocities in the elements within the range of direction K, with K=1 to 36; (m/s).

DVM(K) (NT): mean flow velocity (unweighted) in all the elements within the range of direction K, with K=1 to 36; (m/s).

DWTVM(K) (NT): weighted average flow velocity in all the elements within the range of direction K, with K=1 to 36; (m/s). The weighting factor for the velocity in each element is the product ELLEN x VEL (see DLVEL).

E(N) (NF): conductance of element N, defined as the element conductivity x (cross-section/ length), i.e., (W^2 x FLUID) x [(W x 1)/ELLEN]; (m^2/s).

EFF(L) (NW): "effectiveness" of intersection L; logical variable taking value (.TRUE.) if the intersection L can contribute to flow, and (.FALSE.) if it cannot.

EFFLE(I) (NW): total length of "effective" segments in fracture set I; (m).

EFFVO(I) (NW): total "volume" (i.e., ELLEN x AP x 1) of "effective" segments in fracture set I; (m^3).

ELLEN(N) (NW,AP,NF):	length of element N; (m).
ELOR(N) (NW,AP,NF):	angle of the element N within the simulation plane; same convention as for ATH(I).
ELORP(N) (NW,AP):	angle between the simulation plane and the element N; same convention as for ATHP(I).
ESTNFR (NW):	estimated number of fractures required for the set being processed.
FLUID (NF):	constant equals to the weight density [F/L^3] divided by (12 x the dynamic viscosity [FT/L^2]); (m^{-1} s^{-1}).
FR (NF):	flow rate in the element being considered; positive in the direction of increasing y, because of the convention used to define the elements in subroutine ELTDEF of the program NETWRK; (m^3/s).
I (NW,AP):	index for the fracture sets; equivalent to JSET.
I (NW,NF,NT):	also index for the boundary segments for the uniform random numbers R (NT).
IAAP (NW,AP):	counter of the number of different values of the distribution parameter AAP.
IAP (NW,AP):	counter of the number of realizations of aperture distributions with the same value of AAP.
IB(M) (NW,NF):	code for the location of node M; (0) for an internal node, (1) to (8) mean that the node is located on the corresponding boundary segment.
IBC1 (NT):	boundary condition for boundary segment 1; equals to IBC(1).
IDIFF (NF):	bandwidth of matrix A.
IDIR(J) (NT):	direction (1 to 36) that a particle shall take for the corresponding value of random J (maximum J equals 100).
INC1(L,1) (NW):	first fracture set incidence of node L.
INC1(L,2) (NW):	first fracture incidence of node L; note that this fracture belongs to the set INC1(L,1).

INC2(L,1) (NW):	second fracture set incidence of node L.
INC2(L,2) (NW):	second fracture incidence of node L; note that this fracture belongs to the set INC2(L,1); note also that INC2(L,1 and 2) = 0 for the nodes located on the boundaries.
ISET(N) (NW,AP,NF):	fracture set that contains the element N.
J (NW):	index for the fracture number; equivalent to JFRA.
JO (NW):	number of fractures totally in the buffer margin (X1(J,I)=999.) encountered thus far in the set being processed.
JAP (NF):	counter for the number of realizations of aperture distributions; the value of JAP is within the range 1 to (NAP x NAAP).
JFRA and JJFRA (NW):	index for the fracture number; equivalent to J (in subroutine CIRBDR).
JMEM(M) (NF):	number of nodes related to node M ($1 \leq$ JMEM(M) ≤ 4).
JOINT(MAXNOD) (NF):	work array used in the node renumbering subroutine OPTNUM (see also NEWJT).
JSET (NW):	index for the fracture set; equivalent to I.
JT(2xMAXELT) (NF):	single array equivalent to the two arrays NOD1 (MAXELT) and NOD2 (MAXELT) attached one at the end of the other; JT is used in SETUP and in OPTNUM only.
K (NF,NT):	index for the range of direction; the value of K is between 1 and 18 or between 1 and 36 (degree ÷ 10). K refers to the range of direction from 10(K-1) to 10K.
KDIAG (MAXFRE) (NF):	secondary array required for the variable-bandwith storage of A; it gives the address of all the diagonal elements in the submatrix A.
L (NW):	index for the fracture intersections; equivalent to LINT.
LC(MAXNOD) (NF):	condensation code used in partitioning the head and the node conductance matrices; LC(I) specifies the number of

	constrained nodes that have an index smaller than or equal to I.
LINT (NW):	index for the fracture intersections; equivalent to L.
M (NW,NF):	index for the nodes, or "effective" intersections.
MAXA (NF):	maximum number of entries in the matrix A.
MAXCON (NF):	maximum number of constrained nodes (i.e., with specified head) for which memory space is provided (MAXCON = MAXNOD-MAXFRE).
MAXELT (NW,AP,NF):	maximum number of elements for which memory space is provided.
MAXFRA (NW):	maximum number of fractures for which memory space is provided for a fracture set.
MAXFRE (NF):	maximum number of free nodes (i.e., with non-specified head) for which memory space is provided.
MAXINT (NW):	maximum number of intersections for which memory space is provided.
MAXNOD (AP,NF):	maximum number of nodes (or "effective" intersections) for which memory space is provided.
MEMJT(4(M-1)+1 to 4(M-1)+4) (NF):	node numbers of the nodes related to node M: their number varies between 1 and 4 (see JMEM); the total dimension of MEMJT is 4 x MAXNOD.
MEND (NW):	record of which extremity (either 1 or 2) is being trimmed in the fracture trace being processed; it is used only in the subroutines CHOPXY, CHOPC and BNODE.
N (NW,AP,NF):	index for the elements.
NAPER (NF,NT):	total number of different realizations of aperture distribution; NAPER=NAP x NAAP.
NBN (NW):	number of nodes located on a boundary for the fracture set being processed.
NBO (NW,NF):	number of different boundary segments in the model, NBO is 6 for a circular model and 8 for a rectangular model.

ND1 and ND2 (AP):	number of the first node and of the second node of the element being processed (see NOD1 and NOD2).
NEWJT(MAXNOD) (NF):	work array used in the node renumbering subroutine OPTNUM (see also JOINT).
NEWNOD(M) (NF):	after node renumbering, it specifies the old node number corresponding to the new node M (see NODNUM).
NFRAC(I) (NW):	number of fractures in the set I at the end of the generation process.
NOD1(N) & NOD2(N) (NW,AP,NF):	number of the first node (lower y-value) and the second node (higher y-value) of the element N (see also JT).
NODNUM(M) (NF):	after node renumbering, it specifies the new node number corresponding to the old node M (see NEWNOD).
NSEG(K) (NF,NT):	number of elements in the range of direction K, with K=1 to 36.
NT (NT):	index for the particle being tracked.
NUMELT (NW,AP,NF):	number of elements in the model.
NUMINT (NW):	total number of fracture intersections in the model, including the intersections of fractures with boundaries.
NUMNPT (NW,AP,NF):	number of nodes, or "effective" intersections, in the model.
P(N) (AP):	first direction cosine of element N; see also Q(N) and R(N); the conventions are given in Figure D.2.
PHI(M) (NF):	hydraulic head value at node M; (m). In subroutine SOLPHI, PHI is used as a work array to store various vectors.
PHIC(MAXCON) (NF):	vector of constrained (i.e., specified) hydraulic head values.
Q(N) (AP):	second direction cosine of element N; see also P(N) and R(N).
R(N) (AP):	third direction cosine of element N; see also P(N) and Q(N).
R(100) (NT):	array of uniformly distributed random numbers.

RDFR(K) (NT):	ratio of the sum of flow rate in the range of direction K, i.e., DFR(K), over the total sum of flow rates in all the directions.
RF (AP):	radial coordinate of the centre point of the element being processed, from the centre (0,0) of the model; (m).
SPGR (NF):	weight density of the fluid; (N/m^3).
SUMLE (NW):	sum of the lengths of the fracture traces generated thus far for the set being processed.
TANTH(J,I) (NW):	equivalent to TH(J,I).
TH(J,I) (NW):	trace angle of fracture J in set I, within the simulation plane; same convention as for ATH(I); it is expressed as the tangent in most of the program NETWRK and in radiants in the output.
THP(J,I) (NW):	angle between the simulation plane and fracture J in set I; same convention as for ATHP(I).
THR (AP):	angular coordinate of the centre point of the element being processed; same convention as for ATH(I).
TIM(NT) (NT):	total travel time for the particle NT to go through the specified distance TRAVDIS (s).
TRACE(J) (NW):	trace length of fracture J in the set being processed; (m).
TRAV(NT) (NT):	cumulative absolute distance travelled by particle NT; (m).
VEL (NF):	fluid velocity in the element being considered; positive in the direction of increasing y, like FR; (m/s).
VISC (NF):	dynamic viscosity of the fluid; (kg/(m.s)).
W(N) (NW,AP,NF):	aperture (width) of element N; (m).
X1(J,I), Y1(J,I) (NW):	coordinates of extremity 1 (lower y value) of fracture J in set I; (m). Note that X1(J,I) = 999.0 if the fracture

	(J,I) is entirely within the buffer margin.
X2(J,I), Y2(J,I) (NW):	coordinates of extremity 2 (higher y value) of fracture J in set I; (m).
X1E,Y1E and X2E,Y2E (AP):	coordinates of the extremities 1 (lower y value) and 2 (higher y value) of the element being processed; (m).
XC(J) and YC(J) (NW):	coordinates of the centre point of the trace of fracture J in the set being processed; (m).
XORD(L, or M) and YORD (L or M) (NW,AP,NF):	coordinates of the intersection L or of the node M; (m).
XTRAV(NT) and YTRAV(NT) (NT):	total distance travelled by particle NT in the x-direction and in the y-direction; (m).
Y1(J,I) (NW):	see X1(J,I).
Y2(J,I) (NW):	see X2(J,I).
Y1E and Y2E (AP):	see X1E.
YC(J) (NW):	see XC(J).
YORD(L or M) (NW,AP,NF):	see XORD (L or M).
YTRAV(NT) (NT):	see XTRAV(NT).

Appendix B
Input/Output Files

Input/Output Files

Table B.1. List of All the Input/Output Files

Unit no.	Mnemonic name	Description
1	INP1	General input data.
2	SUMNET	Summary information on the generated line network, on fracture apertures (from the program NETWRK) and on porosity.
2+	SUMNET	All the above plus information on individual fracture traces.
3	SPACNG1	Spacing data generated by the subroutine SPCNG1.
4	SPCNG2	Spacing data generated by the subroutine SPCNG2.
5	NODES	Unformatted file containing node data.
6	ELEM	Unformatted file containing element data.
7	APER1	Unformatted file containing aperture data generated by the program NETWRK.
15	NODFO	Formatted equivalent of the file No. 5.
16	ELEFO	Formatted equivalent of the file No. 6.
17	APEFO1	Formatted equivalent of the file No. 7.
21	INP2	Input data on aperture distribution for the program APEGEN.
22	APER2	Unformatted file containing aperture data generated by the program APEGEN.

Table B.1. Continued

Unit no.	Mnemonic name	Description
23	APEF02	Formatted equivalent of the file No. 22.
31	SUMFLO	Summary of the flow calculations, including the directional parameters.
31+	SUMFLO	The above plus rose diagrams of flow rates.
31++	SUMFLO	The above plus rose diagrams of all the directional parameters.
32	DIRPAR	Unformatted file of directional parameters.
33	FLOALL	Results of flow calculations for all the elements.
34	FLOMAT	Detailed information on node renumbering and on flow matrices.
41	TRANSIT	Statistics and plots of transit times.

Table B.2. Input Files and Options of Output Files for Each Program

Program	Input unit no.	Output	
		IPRT	Unit no.
NETWRK*	1	1	2,3,4,5,6,7
		2 or 3	2+,3,4,5,6,7, 15,16,17
APEGEN	21,5,6,7	1	22
		2 or 3	22,23
NETFLO	1,5,6,7**	1	31,32
		2	31+,32,33
		3	31++,32,33,34
NETRANS	1,32	1, or 2, or 3	41

* NETWRK also generates a plotting file if the input parameter PLTSIZ is positive.

** In NETFLO, the input unit 7 must correspond to either the output unit 7 from NETWRK (APER1) or the output unit 22 from APEGEN (APER2).

Appendix C
Format of Input Data

Format of Input Data

C.1. General Input File (Unit 1)

. Record 1

- Columns 1 to 80: TITLE (20A4).

. Record 2: general information.

- Columns 1 to 5: NSET (I5).
- Columns 6 to 10: ISP (I5).
- Columns 11 to 15: COEF (F5.0).
- Columns 16 to 20: SEED (F5.0).
- Columns 21 to 25: PLTSIZ (F5.0).
- Columns 26 to 30: IGEOM (I5).
- Columns 31 to 35: NAP (I5).
- Columns 36 to 40: NAAP (I5).
- Columns 41 to 50: DAAP (F10.0).
- Columns 51 to 55: IPRT (I5).
- Columns 56 to 60: NPART (I5).
- Columns 61 to 65: TRAVDIS (F5.0).
- Columns 66 to 68: NMAP (I3).

. Record 3: location of the rectangular boundaries. These are required even for circular model.

- Columns 1 to 40: XMINO, YMAXO, XMAXO, YMINO, XMINI, YMAXI, XMAXI, YMINI (8F5.0). (in that clockwise order)

. Record 4 and following: other information on model boundaries. One record for each on the NBO boundaries: for a rectangular model there are 8 boundaries, for a circular model there are 6 boundaries (I = 1 to NBO).

- Columns 1 to 5: ISHAP(I) (I5).
- Columns 6 to 10: R(I) (F5.0).
- Columns 11 to 15: XB(I) (F5.0).
- Columns 16 to 20: YB(I) (F5.0).
- Columns 21 to 25: IBC(I) (I5).
- Columns 26 to 30: HDB(I) (F5.0).
- Columns 31 to 35: HDE(I) (F5.0).

. Following records: first series of information on the fracture network. One record for each one of the NSET fracture sets (I = 1 to NSET).

- Columns 1 to 10: DENS(I) (F10.0).
- Columns 11 to 15: LOSTR(I) (I5).
- Columns 16 to 20: THDSTR(I) (I5).
- Columns 21 to 25: APDSTR(I) (I5).
- Columns 26 to 30: ITERM(I) (I5).

. Following records: parameters of the statistical distributions for the fracture network. One record for each one of the NSET fracture sets (I = 1 to NSET).

- Columns 1 to 10: ALE(I) (F10.0).
- Columns 11 to 20: BLE(I) (F10.0).
- Columns 21 to 30: ATHP(I) (F10.0).
- Columns 31 to 40: BTHP(I) (F10.0).
- Columns 41 to 50: ATH(I) (F10.0).
- Columns 51 to 60: BTH(I) (F10.0).
- Columns 61 to 70: AAP(I) (F10.0).
- Columns 71 to 90: BAP(I) (F10.0).

. Last record:

array of NMAP particle numbers for which the detailed trajectory is to be printed (I = 1 to NMAP)

- Columns 1 to 30: MPART(I) (10I3).

C.2 Supplementary Input File for the Program APEGEN (Unit 21)

. Record 1

- Columns 1 to 80: TITLE (20A4).
  (It is suggested to make this title similar to the title on the general input file, and to add an identification for the aperture generation run.)

. Record 2: general information.

- Columns 1 to 5: IPRT (I5).
- Columns 6 to 10: APDSTR (I5).
- Columns 11 to 15: NAP (I5).
- Columns 16 to 20: NAAP (I5).
- Columns 21 to 30: DAAP (F10.0).
- Columns 31 to 35: SEED (F5.0).
- Columns 36 to 40: NSET (I5).
- Columns 41 to 45: IOR (I5).

- Record 3 and the following: parameters of aperture distributions. One record for each one of the NSET fracture sets (I = 1 to NSET).

  - Columns 1 to 10: AAP(I) (F10.0).
  - Columns 11 to 20: BAP(I) (F10.0).

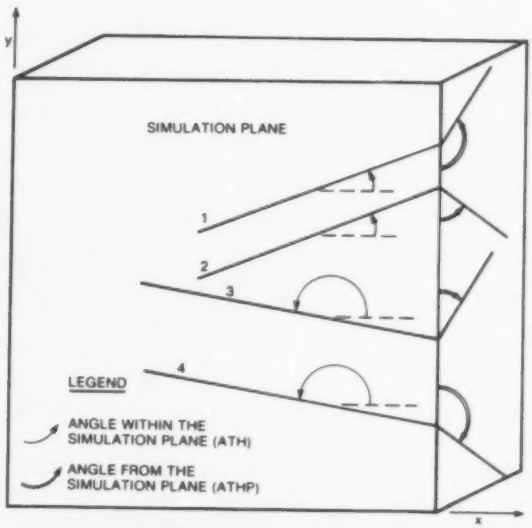
Appendix D

Conventions for Angles, Directions, and Boundary Geometry

Conventions for Angles, Directions, and Boundary Geometry

D.1 Conventions for Angles

Figure D.1 explains the convention used to express the angles within the simulation plane (ATH, TH, and ELOR) and the angles from the simulation plane (ATHP, THP, and ELORP).



ORIENTATION DATA

FRACTURE	ATH (degrees)	ATHP (degrees)
1	20	135
2	20	45
3	170	45
4	170	135

Figure D.1. Conventions for angles.

D.2 Conventions for the Direction Cosines Computed in the Program APEGEN

The program APEGEN offers the possibility of generating aperture values that are functions of the fracture orientation (Sec. 2.9). For that purpose, APEGEN computes the direction cosines of every fracture segment (or flow element) using the conventions shown in Figure D.2.

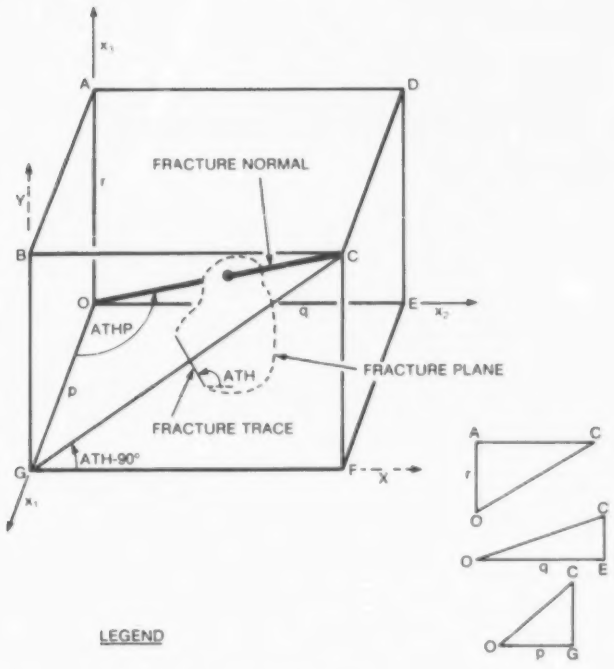


Figure D.2. Conventions for the direction cosines computed in the program APEGEN.

The angles TH and THP of the fracture in Figure D.2 correspond to the angles ATH and ATHP respectively in Figure D.1, and to the angles ELOR and ELORP for the fracture segments in the program APEGEN.

In Figure D.2, OC is the unit vector normal to the fracture plane. The three direction cosines are defined as:

$$p = \cos GOC = OG$$
$$q = \cos EOC = OE = BC$$
$$r = \cos AOC = OA = CF$$

Now, we must express p, q, r in terms of the angles TH and THP. From Figure D.2, p is simply cos THP. Also, we know that:

$$\sin BGC = BC/GC$$

and

$$\sin GOC = GC$$

Then
$$q = BC = GC \sin BGC = \sin GOC . \sin BGC$$
$$q = \sin THP . \sin(90-(TH-90))$$
$$q = \sin THP . \sin(180-TH)$$
$$q = \sin THP . \sin TH$$

We also know that:

$$\sin CGF = CF/GC$$

Then
$$r = CF = GC \sin CGF = \sin GOC . \sin CGF$$
$$r = \sin THP . \sin (TH-90)$$
$$r = -\sin THP . \cos TH$$

In summary:
$$p = \cos THP$$
$$q = \sin THP . \sin TH$$
$$r = -\sin THP . \cos TH.$$

D.3. Conventions and Limitations on Boundary Geometry

1. Every non-circular side of a model may be divided in two segments, each one having a different type of boundary condition.

2. Flow boundaries are numbered clockwise, starting with the vertical boundary on the left-hand side of a model (Fig. D.3). A rectangular model must have 8 flow boundaries, a circular model must have 6. In a rectangular model, flow-boundary numbers 1, 3, 5, and 7 must always be present (i.e., ISHAP(I) > 0). Flow-boundary numbers 2, 4, 6, and 8 may have a zero length (i.e., ISHAP(I)=0). In a circular model, the compulsory flow-boundaries are numbers 1, 3, 4, and 6; the optional boundaries are 2 and 5. Note that if the flow-boundary I has zero length (i.e., ISHAP(I)=0), the flow boundary numbered I-1 must apply to the entire length of the side of the model where it is located.

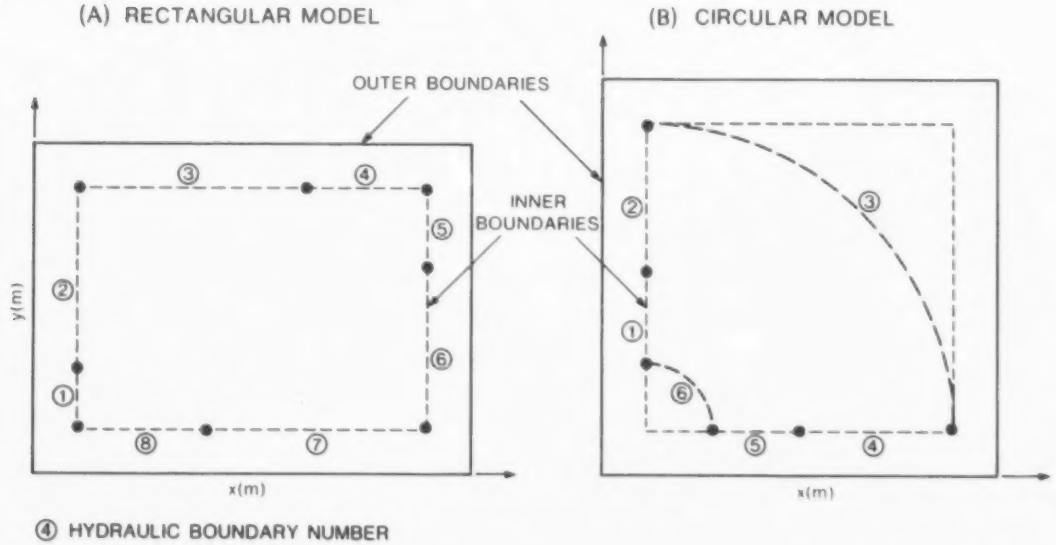


Figure D.3. Numbering of the flow boundaries: (A) rectangular model, (B) circular model.

3. The input mean orientation (ATH) should be within the range 0° to 180°.

4. All rectangular boundaries (i.e., XMAXI...etc., and XMAXO...etc.) must be included in input data even for a circular model.

5. A circular boundary must correspond to the upper right quadrant of a circle and be centred at the point (XMINI, YMINI).

6. The present version of NETRANS can be used only for rectangular models.

Appendix E
Array Dimensioning

Array Dimensioning

The following four pages reproduce the beginning of the four programs (NETWRK, APEGEN, NETFLO, and NETRANS) and of the subroutine TTIMER in NETRANS. Arrows indicate the statements where changes may be made if array redimensioning is desired. Note that the program NETRANS (and the subroutine TTIMER) should not require any change in array dimensions (see Sec. 4.2.3). If one wishes to change the maximum number of particles, however, the value of MAXNPRT must be changed in the main program (NETRANS), as well as the dimension of the following arrays in the subroutine TTIMER: XTRAV, YTRAV, TIM, NSTEP, and TRAV.

```
      PROGRAM NETWRK
C*****************************************************************************
C
C  PROGRAM TO GENERATE A TWO-DIMENSIONAL NETWORK OF FINITE LINE SEGMENTS
C  AND TO COMPUTE SEVERAL STATISTICS ON THE GENERATED LINE NETWORK.
C  (WRITTEN BY A. ROULEAU, 1983; UPDATE BY A. ROULEAU, JAN. 1986)
C
C*****************************************************************************
C
      DOUBLE PRECISION SEED,PI,RAD
      INTEGER THDSTR,APDSTR
C
C DIMENSION NSET = 4 , AND MAXFRA = 100
      DIMENSION THP(100,4)
      DIMENSION X1(100,4),Y1(100,4),X2(100,4),
     *Y2(100,4),TH(100,4),TANTH(100,4)
      INTEGER CUT1(100,4),CUT2(100,4)
C
C DIMENSION MAXFRA = 100
      DIMENSION XC(100),YC(100),TRACE(100),DEGRE(100),
     *DEGREP(100),APER(100)
C
C DIMENSION MAXINT=150; 2*MAXINT=300 (SEE DATA STATEMENT)
      DIMENSION XORD(150),YORD(150)
      INTEGER INC1(150,2),INC2(150,2),
     *IB(150)
      LOGICAL EFF(150)
C
C DIMENSION MAXELT = 200
      DIMENSION ELOR(200),ELORP(200),W(200),ELLEN(200)
      INTEGER NOD1(200),NOD2(200),ISET(200)
C
      COMMON /AREA1A/ XMINO,YMAXO,XMAXO,YMINO
      COMMON /AREA1B/ XMINI,YMAXI,XMAXI,YMINI,IGEOM
      COMMON /AREA2/ NSET,MAXFRA,MAXINT,MAXELT,NUMINT,NUMNPT,NUMELT,
     *NOINT,IPRT
      COMMON /AREA3A/ SEED,PI,RAD,COEF
      COMMON /AREA3B/ TITLE(20)
      COMMON /AREA4/ ATH(5),BTH(5),ATHP(5),BTHP(5),THDSTR(5),NFRAC(5)
      COMMON /AREA5/ DENS(5),ALE(5),BLE(5),LDSTR(5),ITERM(5)
      COMMON /AREA6/ AAP(5),BAP(5),APDSTR(5),DAAP,NAAP,IAAP,NAP,IAP
      COMMON /AREA7/ R(8),XB(8),YB(8),ISHAP(5),IBC(8)
C
      EQUIVALENCE (TH,TANTH),(DEGRE,APER),(TRACE,EFF,NOD1)
      EQUIVALENCE (IB,NOD2),(W,XC)
C
C DIMENSION MAXINT=150  (2*MAXINT=300)
      DATA IB/150*0/,INC2/300*0/
C
C ASSIGN VALUES TO SOME CONSTANTS
      MAXFRA=100
      MAXINT=150
      MAXELT=200
      PI=3.1415926536
      RAD=PI/180.
```

```
      PROGRAM APECEN
C*****************************************************************************
C
C  PROGRAM APECEN GENERATES FRACTURE APERTURE VALUES FOR THE FRACTURE
C  NETWORK GENERATED BY THE PROGRAM NETWRK
C
C  WRITTEN BY A. POULEAU, DECEMBER 1984; UPDATE BY A. POULEAU
C  AND R. MACLEOD, JUNE 1985.
C
C*****************************************************************************
      INTEGER APDSTR
      DOUBLE PRECISION SEED
      COMMON /AREA1/ TITLE(20)
      COMMON /AREA2/ NUMNPT,NUMELT,IPRT
      COMMON /AREA3/ AAP(5),BAP(5),SEED,APDSTR,NSET
      COMMON /AREA4/ DAAP,NAAP,MAP,IAAP,IAP
C
C DIMENSION MAXELT=200  <----
      INTEGER NOD1(200),NOD2(200),ISET(200)  <----
      DIMENSION W(200),WW(200),ELLEN(200),ELOR(200),ELORP(200)  <----
      DIMENSION P(200),Q(200),R(200)  <----
C
C DIMENSION MAXNOD=150  <----
      DIMENSION XORD(150),YORD(150)  <----
C
C CONSTANTS
      MAXELT=200  <----
      MAXNOD=150  <----
C
      READ(21,3) TITLE
3     FORMAT(20A4)
C
```

```
      PROGRAM NETFLO
C*****************************************************************
C
C  THE PROGRAM NETFLO IS A NUMERICAL CODE THAT SOLVES THE STEADY-
C  STATE EQUATIONS FOR FLUID FLOW IN THE NETWORK OF FRACTURES GENERATED
C  BY THE PROGRAM NETWRK
C
C  WRITTEN BY A. ROULEAU, 1983; UPDATE BY A. ROULEAU AND D. LENTZ,
C  APRIL 1985.
C
C*****************************************************************
C
      COMMON /AREA2/ MAXNOD,MAXELT,NUMNPT,NUMELT,NC,NF,NA,MAXA,
     *MAXCON,MAXFRE,IPRT,JAP
      COMMON /AREA4/ XB(8),YB(8),HDB(8),HDE(8),ISHAP(8),IBC(8),IGEOM,NBO
      COMMON /AREA5/ MAXEL2,MAXNO4,IDIFF
      COMMON /AREA6/ TITLE(20)
      COMMON /AREA8/ VISC,SPGR
C
C DIMENSION = MAXELT = 200
      INTEGER NOD1(200),NOD2(200),IDUMMY(200)
      DIMENSION W(200),E(200),ELLEN(200),ELOR(200)
C
C DIMENSION = MAXEL2 = 2 * MAXELT = 400
      INTEGER JT(400)
C
C DIMENSION = MAXNOD = 150
      INTEGER IB(150),LC(150),JMEM(150),NODNUM(150),NEWNOD(150),
     *JOINT(150),NEWJT(150)
      DIMENSION XORD(150),YORD(150),PHI(150)
      LOGICAL REOR(150)
C
C DIMENSION = MAXNO4 = 4 * MAXNOD = 600
      INTEGER MEMJT(600)
C
C DIMENSION = MAXFRE = 110
      INTEGER KDIAG(110)
C
C DIMENSION = MAXCON = MAXNOD - MAXFRE = 150-110=40
      DIMENSION PHIC(40)
C
C DIMENSION = MAXA = 2000
      DIMENSION A(2000)
C
C MAXELT + 1 = 200 + 1 = 201
      EQUIVALENCE (JT(1),NOD1),(JT(201),NOD2)
      EQUIVALENCE (ELOR,IDUMMY)
C
C CONSTANTS
C
      MAXELT = 200
      MAXNOD = 150
      MAXA = 2000
      MAXFRE = 110
      MAXCON = MAXNOD-MAXFRE
```

```
      PROGRAM NETPAR5
C*****************************************************************************
C
C  PROGRAM TO SIMULATE STOCHASTICALLY THE MIGRATION OF PARTICLES
C  THROUGH A FRACTURE NETWORK BASED ON STATISTICS OF THE DIRECTIONAL
C  PARAMETERS
C
C  WRITTEN BY A. ROULEAU AND D. LENTZ, 1984; UPDATE BY A.ROULEAU, JANUARY 1986.
C
C*****************************************************************************
      COMMON /AREA1/ COSTH(36),SINTH(36),DLM(36),DWTVM(36),PDFR(36),
     +XLEN,YLEN,TRAVDIS,IDIR(100),MPART(10),IBC1,IPRT,JAP
      COMMON/AREA2/TITLE
      DIMENSION DVEL(36),DFR(36),DLE(36),DLVEL(36),DLVEL2(36),NSEG(36)
      CHARACTER TITLE(20)*4,FMT(5)*20
      DATA (FMT(I),I=1,5)/'(2(/),10I3)','(4(/),10I3)','(6(/),10I3)',
     *'(8(/),10I3)','(10(/),10I3)'/
      MAXNPRT=50
C
C**** ***************************************************** ********
C
C     SUBROUTINE TTIMER COMPUTES STOCHASTICALLY THE TRANSIT TIME OF
C  PARTICLES THROUGH THE NETWORK USING THE STATISTICS OF THE
C  DIRECTIONAL PARAMETERS (RECTANGULAR MODEL)
C
C*****************************************************************************
      SUBROUTINE TTIMER(NPART,NM)
      COMMON /AREA1/ COSTH(36),SINTH(36),DLM(36),DWTVM(36),PDFR(36),
     +XLEN,YLEN,TRAVDIS,IDIR(100),MPART(10),IBC1,IPRT,JAP
      COMMON/AREA2/TITLE
      DIMENSION R(100),XTRAV(50),YTRAV(50),TIM(50)
     *,NSTEP(50),TRAV(50)
      DOUBLE PRECISION DSEED
      CHARACTER TITLE(20)*4,BCD1*22
C
```

Appendix F
Sample Runs

Sample Runs

This appendix reproduces the input files and portions of the output files for three sample runs of the four programs NETWRK, APEGEN, NETFLO, and NETRANS. The first sample run (realization 990) shows a rectangular model without abutting fractures. Realization 991 uses the circular geometry. Realization 992 shows a rectangular model again, but this time with abutting fractures in sets 2, 3, and 4.

F.1 A Rectangular Model Without Abutting Fractures: Realization 990

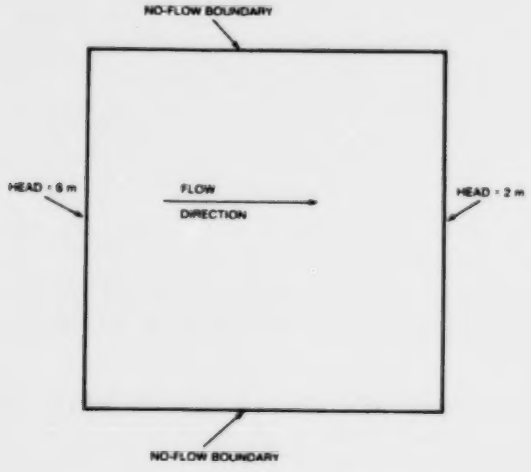


Figure F.1. Boundary conditions for realization 990.

File INP1 — unit 1

```
990 SAMPLE RUN OF SDF PACK. : RECTANGULAR MODEL
   4     1  3.0 990.  25.    1    1    1                2   50 100.  0
-1.0 11.0 11.0 -1.0  0.0 10.0 10.0  0.0
   1        0.0  0.0    1  6.0
   0
   1        0.0 10.0    0
   0
   1       10.0 10.0    1  2.0
   0
   1       10.0  0.0    0
   0
     0.40    2    2    1    0
     0.60    2    2    1    0
     0.80    2    2    1    0
     1.00    2    2    1    0
      1.4       0.80      15.0       10.0      165.0       10.0  0.000005       0.0
      1.2       0.80      65.0       10.0      125.0       10.0  0.000005       0.0
      1.0       0.80     125.0       10.0       65.0       10.0  0.000005       0.0
      0.8       0.80     165.0       10.0       15.0       10.0  0.000005       0.0
1 13 44
```

File INP2 — unit 21

```
R990-1 FIRST SET OF APER. DATA FOR SAMPLE RUN ON RECT. MODEL
   2    2    1    3        0.5 990.    4    0
    -13.0       0.8
    -12.0       0.8
    -11.0       0.8
    -10.0       0.8
```

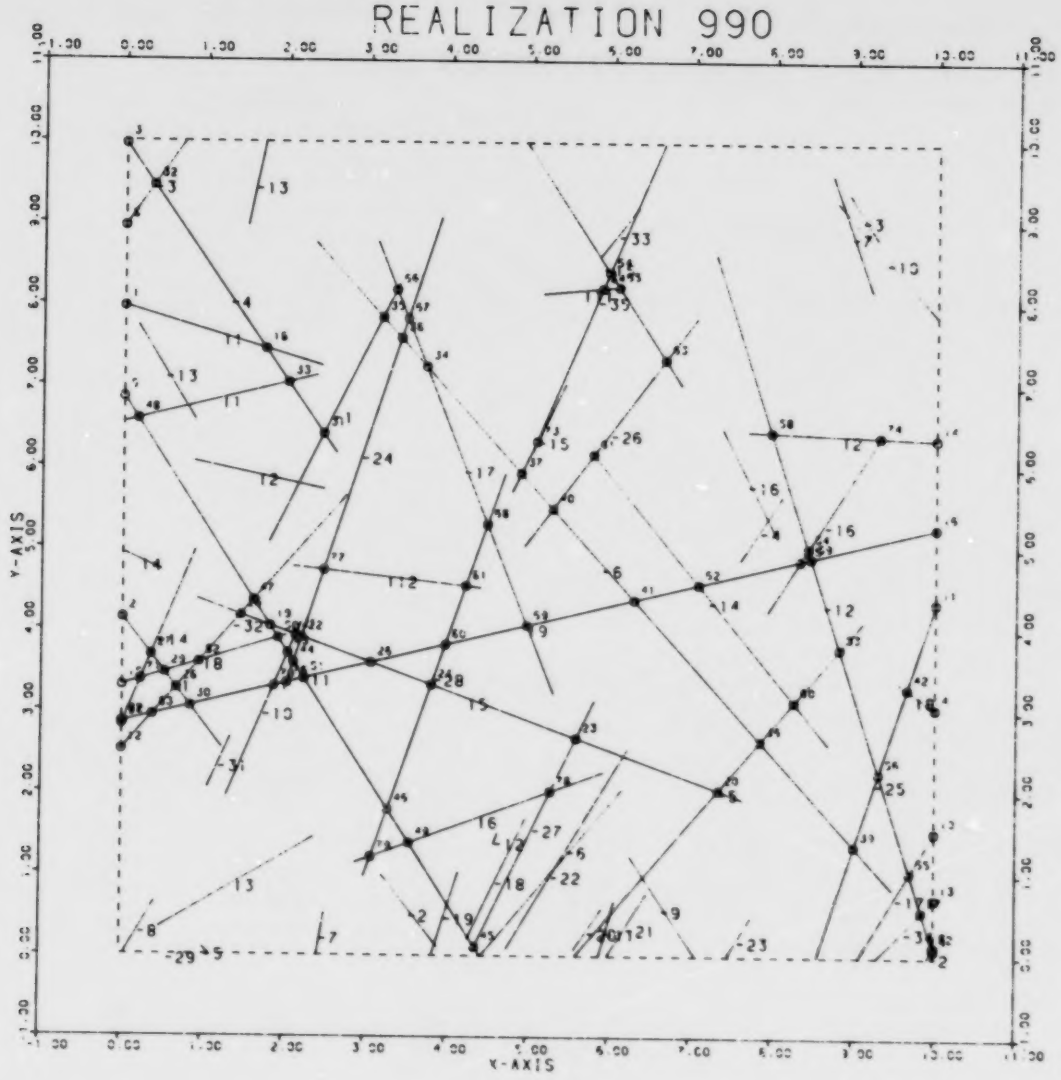


Figure F.2. Computer plot for realization 990. The numbers smaller in size refer to the nodes, the larger numbers refer to the fractures but without indication of which fractures set; these numbers are not printed when NUMNPT>200 or IPRT = 1.

File SUMNET - unit 2

P990 SAMPLE RUN OF SDF PACK. : RECTANGULAR MODEL
FILE: SUMNET.U02

INPUT BOUNDARY DATA

NUMBER	SHAPE	RADIUS	XB	YB	IBC
1	1	.00	.000	.000	1
2	0	.00	.000	.000	0
3	1	.00	.000	10.000	0
4	0	.00	.000	.000	0
5	1	.00	10.000	10.000	1
6	0	.00	.000	.000	0
7	1	.00	10.000	.000	0
8	0	.00	.000	.000	0

INPUT DATA FOR FRACTURE NETWORK

FRACTURE SET :	1	2	3	4
DENSITY :	.400	.600	.800	1.000
DISTRIBUTION TYPE				
FOR TRACE LENGTH :	2	2	2	2
FOR ORIENTATION :	2	2	2	2
DISTRIBUTION PARAMETERS				
FOR TRACE LENGTH				
A :	1.40	1.20	1.00	.80
B :	.80	.80	.80	.80
FOR ORIENTATION WITHIN SIMULATION PLANE				
A :	165.00	125.00	65.00	15.00
B :	10.00	10.00	10.00	10.00
FOR ORIENTATION FROM SIMULATION PLANE				
A :	15.00	65.00	125.00	165.00
B :	10.00	10.00	10.00	10.00

FRACTURES OF SET 1

NUMBER GNRTD	NUMBER ACTUAL	LENGTH	ANGLE	X1	Y1	X2	Y2
	1	2.543	163.437	.000	7.965	2.437	7.240
	2	1.644	168.086	.864	6.060	2.473	5.720
	3	3.084	167.057	999.000	.691	-.788	.000
	4	.528	155.608	.000	4.934	.481	4.716
	5	7.160	160.073	.909	4.372	7.640	1.932

File SUMNET – unit 2 (continued)

SUMMARY OF FRACTURE SET 1

DENSITY OF FRACTURING .1035 L/L**2
THEORETICAL MEAN TRACE LENGTH 5.9845
SURFACE AREA OF THE MODEL 100.0000
DESIRED TOTAL LENGTH FOR THE SET 10.3528
ACTUAL TOTAL LENGTH FOR THE SET 11.8748
ESTIMATED NUMBER OF FRACTURES 5
ACTUAL NUMBER OF FRACTURES GENERATED 5
NUMBER OF FRACTURES TOTALLY IN THE BUFFER MARGIN 1
NUMBER OF INTERSECTIONS WITH A BOUNDARY 2

FRACTURES OF SET 2

NUMBER GNRTD	ACTUAL	LENGTH	ANGLE	X1	Y1	X2	Y2
1		2.027	127.667	.000	4.140	1.239	2.535
2		1.135	128.303	3.206	.994	3.909	.103
3		.657	122.213	8.897	9.389	9.248	8.833
4		4.519	124.604	.000	9.967	2.623	6.165
5		.139	132.405	1.009	.103	1.103	.000
6		10.676	132.051	2.337	8.760	9.488	.833
7		1.525	109.211	8.691	9.627	9.192	8.187
8		.571	144.857	9.533	9.386	10.000	9.038
9		1.483	123.618	6.299	1.235	7.080	.000
10		1.888	131.471	8.750	9.293	10.000	7.878
11		8.161	122.922	.000	6.850	4.435	.000
12		8.970	107.750	7.265	8.640	10.000	.097
13		1.370	121.505	.166	7.735	.882	6.567
14		4.629	129.336	5.796	6.189	8.690	2.609
15		3.599	123.007	4.923	10.000	6.858	7.021
16		1.465	117.205	7.362	6.493	8.032	5.190
17		5.980	111.680	3.101	8.776	5.310	3.219

SUMMARY OF FRACTURE SET 2

DENSITY OF FRACTURING .5438 L/L**2
THEORETICAL MEAN TRACE LENGTH 4.5722
SURFACE AREA OF THE MODEL 100.0000
DESIRED TOTAL LENGTH FOR THE SET 54.3785
ACTUAL TOTAL LENGTH FOR THE SET 58.8479
ESTIMATED NUMBER OF FRACTURES 35
ACTUAL NUMBER OF FRACTURES GENERATED 17
NUMBER OF FRACTURES TOTALLY IN THE BUFFER MARGIN 0
NUMBER OF INTERSECTIONS WITH A BOUNDARY 10

FRACTURES OF SET 3

NUMBER GNRTD	ACTUAL	LENGTH	ANGLE	X1	Y1	X2	Y2

...etc

File SUMNET - unit 2 (end)

APERTURE AND POROSITY DATA

INPUT DATA FOR APERTURE DISTRIBUTIONS IAAP= 1

FRACTURE SET :	1	2	3	4
DISTRIBUTION TYPE :	1	1	1	1
DISTRIBUTION PARAMETERS				
A :	5.00E-06	5.00E-06	5.00E-06	5.00E-06
B :	.00	.00	.00	.00

NUMBER OF ELEMENTS (NUMELT) : 116

SUMMARY OF POROSITY DATA IAAP= 1 IAP= 1

SET	LENGTH		VOLUME	
	EFFECTIVE	DEAD	EFFECTIVE	DEAD
1	8.67	3.81	4.03E-05	1.90E-05
2	39.18	19.67	1.96E-04	9.83E-05
3	31.58	37.66	1.58E-04	1.88E-04
4	20.53	5.47	1.03E-04	2.73E-05

********** //// END OF LIST ////
********** //// END OF LIST ////

File SPACING1 — unit 3

R990 SAMPLE RUN OF SDF PACK. : RECTANGULAR MODEL
FILE: SPACING1.U03

FRACTURE SET 1

NUMB	NTEN	LINE	XP	YP	FRACT	SPACING
1	1	7	.25	1.19	2	1.08
2	1	7	.25	1.19	5	1.00
3	2	4	1.77	4.76	5	1.14
4	2	8	1.81	4.13	5	1.15
5	4	6	1.13	9.66	4	1.89
6	5	7	1.39	4.62	2	1.10
7	5	7	1.39	4.62	5	.99
8	5	8	3.11	9.81	2	1.12
9	5	8	3.11	9.81	5	.98
10	5	9	1.97	9.85	2	1.02
11	6	4	3.03	9.66	2	1.12
12	6	4	3.03	9.66	5	.98
13	6	6	1.85	9.67	2	1.03
14	6	8	.49	5.26	4	1.92
15	7	4	1.78	8.67	2	1.02
16	7	5	1.39	8.53	4	1.93
17	8	1	.50	5.82	4	1.91
18	9	5	2.97	7.99	5	1.17
19	9	6	2.66	9.93	2	1.02
20	9	8	.88	7.76	4	1.90
21	9	10	.90	4.73	2	1.01
22	10	2	.64	2.69	2	1.08
23	10	2	.64	2.69	5	1.00
24	10	10	1.40	9.16	4	1.91
25	12	8	.34	.96	2	1.09
26	12	8	.34	.96	5	.99
27	13	1	1.39	4.94	2	1.09
28	13	1	1.39	4.94	5	.99
29	13	2	.99	1.31	5	1.15
30	14	6	.30	1.88	2	1.07
31	14	6	.30	1.88	5	1.00
32	14	8	.44	5.52	4	1.92
33	14	10	.95	2.27	2	1.11
34	14	10	.95	2.27	5	.98
35	15	5	.11	2.97	2	1.02
36	15	7	.90	4.15	2	1.07
37	15	7	.90	4.15	5	1.01
38	15	8	3.12	9.32	5	1.15
39	15	9	1.47	3.65	5	1.15
40	15	10	2.73	7.68	5	1.15
41	16	10	1.12	5.25	2	1.00
42	17	4	.08	.44	2	1.08
43	17	4	.08	.44	5	1.00
44	18	1	.46	.34	2	1.12
45	18	1	.46	.34	5	.98
46	18	3	.68	5.66	4	1.89
47	18	6	2.89	9.01	2	1.12
48	18	8	2.89	9.01	5	.98
49	18	9	1.44	2.36	5	1.16
50	21	1	1.97	7.67	2	1.08

. . . etc

File NODFO - unit 15

R990 SAMPLE RUN OF SDF PACK. : RECTANGULAR MODEL
FILE: NODFO.U15

INTERSECTION DATA

NUMBER OF INTERSECTIONS
EFFECTIVE : 84 (NUMNPT)
NON-EFFECTIVE : 34 (NOINT)
MAXINT= 150

FILE CONTAINS :
XORD(NUMNPT), FORMAT(10F10.6)
YORD(NUMNPT), FORMAT(10F10.6)
IB(NUMNPT), FORMAT(80I2)
INC1(MAXINT,2), FORMAT(20I5)
INC2(MAXINT,2), FORMAT(20I5)

.000000 .000000 .000000 10.000000 .000000 10.000000 10.000000 .000000 .000000 10.000000
10.000000 .000000 10.000000 10.000000 .000000 10.000000 .000000 1.737363 1.818023 7.337093
2.131146 2.206437 5.591844 3.803644 1.454607 3.057432 .354742 .666739 .521605 .836477
2.459313 .351168 2.022156 3.713952 3.168090 7.852433 4.893169 3.399670 9.013976 5.285862
6.293326 9.654153 2.036488 2.107275 4.365382 3.276245 1.614642 .176458 3.542914 1.914441
2.237254 9.991733 8.820281 8.425710 9.699223 9.309294 9.850472 7.968618 8.462836 8.263844
5.782113 7.086290 6.656292 5.958018 6.074981 3.337118 3.478614 4.483662 4.977173 1.867637
.226592 .011057 5.084364 9.301488 8.331626 2.025632 2.475746 5.277655 3.075242 3.972881
4.222935 .946643 .376193 5.874889
7.965372 4.139865 9.966771 3.057648 6.850134 .096961 .136448 8.953638 2.830361 1.537404
4.350513 2.516386 .716847 6.361823 3.303467 5.270401 2.854183 7.448657 4.042244 2.041434
3.928728 3.901434 2.674133 3.922403 4.173992 3.592925 3.680333 3.276172 3.464179 3.056294
6.402271 9.457792 7.035881 7.234345 7.839501 2.646322 5.927036 7.582766 1.358607 5.491686
4.374787 3.301105 3.704830 3.595501 .107912 1.790057 4.356360 6.577598 1.378192 3.893328
3.394752 .122788 3.782347 5.014971 1.036575 2.254695 .564080 6.442903 4.898988 3.129221
6.158089 4.566385 7.332004 8.406950 8.226892 8.182250 7.826324 5.298179 4.056776 3.305444
3.373283 2.856854 6.320194 6.389703 4.867285 3.343619 4.732300 2.016731 1.206047 3.814117
4.540642 3.595138 2.945079 8.208392

1 1 1 5 1 5 5 1 1 5 5 1 5 5 1 5 1 0 0 0

File ELEFO — unit 16

R990 SAMPLE RUN OF SDF PACK. : RECTANGULAR MODEL
FILE: ELEFO.U16

ELEMENT DATA

NUMBER OF ELEMENTS : 116 (NUMELT)

FILE CONTAINS :
NOD1(NUMELT), FORMAT(20I5)
NOD2(NUMELT), FORMAT(20I5)
ISET(NUMELT), FORMAT(50I2)
ELLEN(NUMELT), FORMAT(1P10E10.3)
ELOR(NUMELT), FORMAT(1P10E10.3)
FLORP(NUMELT), FORMAT(1P10E10.3)

```
   18   20   23   24   26   22   21   19   30   28   29   27   31   33   18   32   39   36   41   40
   37   34   38    4   45   49   46   51   44   43   50   19   47   48    6   52   57   55   56   53
   59   54   60   62   63   65   69   68   34   67   31   35   52    8   20   36   60   70   43    9
   72   71   37   75   54   55   76   44   22   77   38   39   56   42   40   61   45   78   79   46
   24   80   81   12   83   28   82   25   57   73   84   48   14   74   79   49   15   71   29   82
   17   72   83   30   70   76   51   26   80   69   41   62   75   59   84   81
    1   23   24   26   22   21   19   25   28   29   27    2   33   18   32    3   36   41   40   37
   34   38   35   42   49   46   51   44   43   50   19   47   48    5   52   57   55   56   53   59
   54   58   62   61   65   64   68   34   67   66   35   66    7   32   36   60   53   43   21   72
   71   27   73   54   74   10   44   22   77   38   67   56   42   11   61   63   78   23   46   24
   80   81   68   83   28   82   25   47   13   84   64   33   74   58   49   78   71   29   82   50
   72   83   30   70   76   51   26   80   69   41   62   75   59   16   65   77

1 1 1 1 1 1 1 1 2 2 2
```

File APEFO1 — unit 17

```
R990 SAMPLE RUN OF SDF PACK. : RECTANGULAR MODEL
FILE: APEFO1.U17


                        FRACTURE APERTURES

NUMBER OF ELEMENTS :            116 (NUMELT)
NUMBER OF DIFF. VAL. FOR FIRST PARAM (AAP) :        1 (NAAP)
NUMBER OF REALIZATIONS FOR EACH VAL. OF AAP :       1 (NAP)

FILE CONTAINS : NAAP*NAP ARRAYS W(NUMELT), FORMAT(1P10E10.3)
5.000E-06 5.000E-06 5.000E-06 5.000E-06 5.000E-06 5.000E-06 5.000E-06 5.000E-06 5.000E-06 5.000E-06
5.000E-06 5.000E-06 5.000E-06 5.000E-06 5.000E-06 5.000E-06 5.000E-06 5.000E-06 5.000E-06 5.000E-06
5.000E-06 5.000E-06 5.000E-06 5.000E-06 5.000E-06 5.000E-06 5.000E-06 5.000E-06 5.000E-06 5.000E-06
5.000E-06 5.000E-06 5.000E-06 5.000E-06 5.000E-06 5.000E-06 5.000E-06 5.000E-06 5.000E-06 5.000E-06
5.000E-06 5.000E-06 5.000E-06 5.000E-06 5.000E-06 5.000E-06 5.000E-06 5.000E-06 5.000E-06 5.000E-06
5.000E-06 5.000E-06 5.000E-06 5.000E-06 5.000E-06 5.000E-06 5.000E-06 5.000E-06 5.000E-06 5.000E-06
5.000E-06 5.000E-06 5.000E-06 5.000E-06 5.000E-06 5.000E-06 5.000E-06 5.000E-06 5.000E-06 5.000E-06
5.000E-06 5.000E-06 5.000E-06 5.000E-06 5.000E-06 5.000E-06 5.000E-06 5.000E-06 5.000E-06 5.000E-06
5.000E-06 5.000E-06 5.000E-06 5.000E-06 5.000E-06 5.000E-06 5.000E-06 5.000E-06 5.000E-06 5.000E-06
5.000E-06 5.000E-06 5.000E-06 5.000E-06 5.000E-06 5.000E-06 5.000E-06 5.000E-06 5.000E-06 5.000E-06
5.000E-06 5.000E-06 5.000E-06 5.000E-06 5.000E-06 5.000E-06 5.000E-06 5.000E-06 5.000E-06 5.000E-06
5.000E-06 5.000E-06 5.000E-06 5.000E-06 5.000E-06 5.000E-06

**********                  //// END OF LIST ////
**********                  //// END OF LIST ////
```

File APEF02 - unit 23

R990-1 FIRST SET OF APER. DATA FOR SAMPLE RUN ON RECT. MODEL
FILE : APEF02.U23

FRACTURE APERTURES

NUMBER OF ELEMENTS : 116 (NUMELT)
DISTRIBUTION TYPE : 2

INPUT DATA FOR APERTURE DISTRIBUTIONS IAAP= 1 IAP= 1

FRACTURE SET :	1	2	3	4
DISTRIBUTION PARAMETERS				
A :	-1.30E+01	-1.20E+01	-1.10E+01	-1.00E+01
B :	.80	.80	.80	.80

ARRAY ISET, FORMAT(50I2)

```
1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 4 4 4 4 4 4 4 4 4 4
4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
```

ARRAY WW, FORMAT(1P10E10.3)

```
1.726E-06 7.523E-06 2.515E-06 1.228E-06 2.350E-06 3.985E-07 3.890E-06 9.165E-07 2.006E-06 9.368E-06
8.109E-06 7.618E-06 1.782E-06 9.602E-06 2.603E-06 1.817E-05 7.023E-06 2.228E-05 3.259E-06 1.517E-05
1.325E-05 4.516E-06 8.818E-06 9.216E-06 1.063E-05 2.053E-06 6.752E-06 1.304E-05 1.868E-06 1.790E-06
1.915E-06 1.194E-05 1.469E-05 4.938E-06 1.086E-05 5.760E-06 4.494E-06 1.027E-05 2.059E-05 4.277E-06
5.322E-06 5.745E-06 5.090E-06 1.272E-05 3.250E-05 3.372E-06 5.441E-05 3.988E-06 2.632E-05 5.011E-06
2.590E-05 1.493E-05 4.256E-06 2.056E-05 3.502E-05 8.458E-06 2.299E-05 6.069E-06 3.106E-05 5.392E-05
4.313E-05 7.731E-06 1.462E-05 5.176E-06 2.201E-05 1.064E-05 1.250E-05 4.324E-06 1.096E-05 1.167E-05
1.384E-05 1.519E-05 2.143E-05 1.485E-05 6.136E-06 2.164E-05 3.612E-05 2.459E-05 3.176E-05 1.930E-05
1.414E-05 4.918E-06 4.021E-06 1.529E-05 2.656E-05 9.868E-06 8.824E-06 3.273E-05 2.198E-05 1.103E-05
1.458E-05 3.011E-05 3.233E-05 3.125E-05 9.171E-05 7.124E-06 3.541E-05 1.554E-05 4.352E-05 2.865E-05
8.896E-05 6.721E-05 9.277E-05 5.282E-05 4.236E-05 7.810E-05 1.609E-04 2.749E-04 3.715E-05 3.626E-05
2.572E-05 4.230E-05 1.320E-04 3.044E-05 8.677E-05 3.222E-05
```

. . . etc

File SUMFLO - unit 31

P990 SAMPLE RUN OF SCF PACK. : RECTANGULAR MODEL
FILE: SUMFLO.U31

OUTPUT FROM SUBROUTINE BOUND

TOTAL NUMBER OF NODES 84 (NUMNPT)
NUMBER OF FREE NODES 67 (NF)
NUMBER OF CONSTRAINED NODES 17 (NC)

BDRY NO	IBC	BEG	END	'GRADIENT'
1	1	.000	10.000	.000
2	0	.000	.000	.000
3	0	.000	10.000	.000
4	0	.000	.000	.000
5	1	.000	.000	.000
6	0	.000	.000	.000
7	0	.000	.000	.000
8	0	.000	.000	.000

OUTPUT FROM SETUP

INITIAL IDIFF= 71

OUTPUT FROM OPTNUM

FINAL IDIFF= 10

RESULTS FROM OPSTOR

SIZE OF MATRIX A (NA): 409

File SUMFLO - unit 31 (continued)

SELECTED RESULTS OF FLOW CALCULATIONS FOR SEGMENTS LOCATED AT A BOUNDARY

BDRY.NO	ELEM.NO	WIDTH(M)	LENGTH(M)	VELOCITY(M/SEC)	FLOW RATE(M**3/SEC)
1	1	5.0000000E-06	1.8125744E+00	-2.3578586E-06	1.1789293E-11
1	12	5.0000000E-06	5.8052746E-01	-2.6959696E-06	1.3479848E-11
1	16	5.0000000E-06	6.1836823E-01	-7.7480072E-07	3.8740036E-12
5	24	5.0000000E-06	4.2294354E-01	-4.6445305E-06	-2.3222653E-11
1	34	5.0000000E-06	3.2467384E-01	-5.2352579E-06	2.6176290E-11
5	35	5.0000000E-06	2.7117867E-02	-3.0076542E-07	-1.5038271E-12
5	53	5.0000000E-06	1.5966943E-02	5.1081266E-07	-2.5540633E-12
1	54	5.0000000E-06	6.1440259E-01	7.7980165E-07	3.8990083E-12
1	60	5.0000000E-06	2.8708182E-02	1.9295495E-06	9.6477475E-12
5	66	5.0000000E-06	5.8420596E-01	2.8731184E-06	-1.4365592E-11
5	74	5.0000000E-06	1.1049289E+00	1.7089283E-06	-8.5446415E-12
1	84	5.0000000E-06	5.7034958E-01	3.3441906E-06	1.6720953E-11
5	89	5.0000000E-06	2.1376720E-01	1.7972861E-06	-8.9864306E-12
5	93	5.0000000E-06	6.9906866E-01	-7.0903772E-06	-3.5451886E-11
1	97	5.0000000E-06	2.3710466E-01	4.6448844E-06	2.3224422E-11
1	101	5.0000000E-06	1.1375387E-02	4.8696241E-06	2.4348120E-11
5	114	5.0000000E-06	1.5813977E+00	7.8861186E-06	-3.9430593E-11

File SUMFLO – unit 31 (continued)

SUMMARY OF FLOW CALCULATIONS FOR JAP= 1

TOTAL FLOW ALONG THE BOUNDARIES

BDRY NO	FLOW (M**3/SEC) (- =>FLOW OUT)
1	1.33159E8603E-10
2	.0C00C0CC00E+00
3	.C0000C0000E+00
4	.CCC0000C0CE+00
5	-1.33159E8603E-10
6	.0C00CC0C0CE+00
7	.0000C00C00E+00
8	.0CCC0C0CCCE+00

DIRECTIONAL POROSITY AND PERMEABILITY

DIRECTION (DEGR/10)	LENGTH (M)	POROSITY (M**2)	PERMEABILITY (M**3)
1	2.01E-01	1.00E-06	5.02E-12
2	1.42E+01	7.10E-05	3.55E-10
3	2.35E+00	1.17E-05	5.87E-11
4	.00E+00	.00E+00	.00E+00
5	4.95E+00	2.47E-05	1.24E-10
6	5.31E+00	2.66E-05	1.33E-10
7	9.13E+00	4.56E-05	2.28E-10
8	1.22E+01	6.10E-05	3.05E-10
9	.00E+00	.00E+00	.00E+00
10	.00E+00	.00E+00	.00E+00
11	6.66E+00	3.33E-05	1.67E-10
12	4.44E+00	2.22E-05	1.11E-10
13	1.89E+01	9.46E-05	4.73E-10
14	8.73E+00	4.36E-05	2.18E-10
15	4.23E-01	2.11E-06	1.06E-11
16	.00E+00	.00E+00	.00E+00
17	8.07E+00	4.03E-05	2.02E-10
18	3.79E+00	1.90E-05	9.48E-11

File SUMFLO - unit 31 (continued)

OTHER DIRECTIONAL PARAMETERS

DIRECTION (DEGR/10)	NSEG	(1)VELOC (M/S)	(2)FL.RA (M**3/S)	(3)LENGTH	(3)X(1)	(3)X(1)**2
1	1	8.35E-07	4.18E-12	2.01E-01	1.68E-07	1.40E-13
2	19	1.21E-04	6.03E-10	1.42E+01	9.68E-05	7.68E-10
3	2	3.91E-06	1.96E-11	2.35E+00	5.56E-06	1.40E-11
4	0	.00E+00	.00E+00	.00E+00	.00E+00	.00E+00
5	9	4.11E-05	2.06E-10	4.95E+00	2.71E-05	1.75E-10
6	6	1.76E-05	8.80E-11	3.85E+00	1.32E-05	5.28E-11
7	12	2.52E-05	1.26E-10	9.13E+00	2.24E-05	8.94E-11
8	11	2.93E-05	1.47E-10	1.08E+01	2.90E-05	1.28E-10
9	0	.00E+00	.00E+00	.00E+00	.00E+00	.00E+00
10	0	.00E+00	.00E+00	.00E+00	.00E+00	.00E+00
11	3	7.41E-06	3.71E-11	2.79E+00	8.00E-06	2.36E-11
12	0	.00E+00	.00E+00	.00E+00	.00E+00	.00E+00
13	0	.00E+00	.00E+00	.00E+00	.00E+00	.00E+00
14	0	.00E+00	.00E+00	.00E+00	.00E+00	.00E+00
15	0	.00E+00	.00E+00	.00E+00	.00E+00	.00E+00
16	0	.00E+00	.00E+00	.00E+00	.00E+00	.00E+00
17	0	.00E+00	.00E+00	.00E+00	.00E+00	.00E+00
18	0	.00E+00	.00E+00	.00E+00	.00E+00	.00E+00
19	0	.00E+00	.00E+00	.00E+00	.00E+00	.00E+00
20	0	.00E+00	.00E+00	.00E+00	.00E+00	.00E+00
21	0	.00E+00	.00E+00	.00E+00	.00E+00	.00E+00
22	0	.00E+00	.00E+00	.00E+00	.00E+00	.00E+00
23	0	.00E+00	.00E+00	.00E+00	.00E+00	.00E+00
24	1	1.23E-06	6.13E-12	1.46E+00	1.79E-06	2.20E-12
25	0	.00E+00	.00E+00	.00E+00	.00E+00	.00E+00
26	2	1.43E-06	7.13E-12	1.39E+00	9.10E-07	9.69E-13
27	0	.00E+00	.00E+00	.00E+00	.00E+00	.00E+00
28	0	.00E+00	.00E+00	.00E+00	.00E+00	.00E+00
29	5	1.40E-05	7.01E-11	3.87E+00	1.64E-05	7.93E-11
30	4	9.58E-06	4.79E-11	4.44E+00	8.98E-06	2.69E-11
31	22	7.14E-05	3.57E-10	1.89E+01	5.65E-05	2.10E-10
32	7	3.58E-05	1.79E-10	8.73E+00	4.52E-05	2.60E-10
33	1	4.46E-06	2.23E-11	4.23E-01	1.89E-06	8.43E-12
34	0	.00E+00	.00E+00	.00E+00	.00E+00	.00E+00
35	8	4.62E-05	2.31E-10	8.07E+00	4.29E-05	2.67E-10
36	3	1.30E-05	6.49E-11	3.79E+00	1.42E-05	6.31E-11

File SUMFLO - unit 31 (end)

ROSE OF FLOW RATES (M**3/S)

//// END OF LIST ////
//// END OF LIST ////

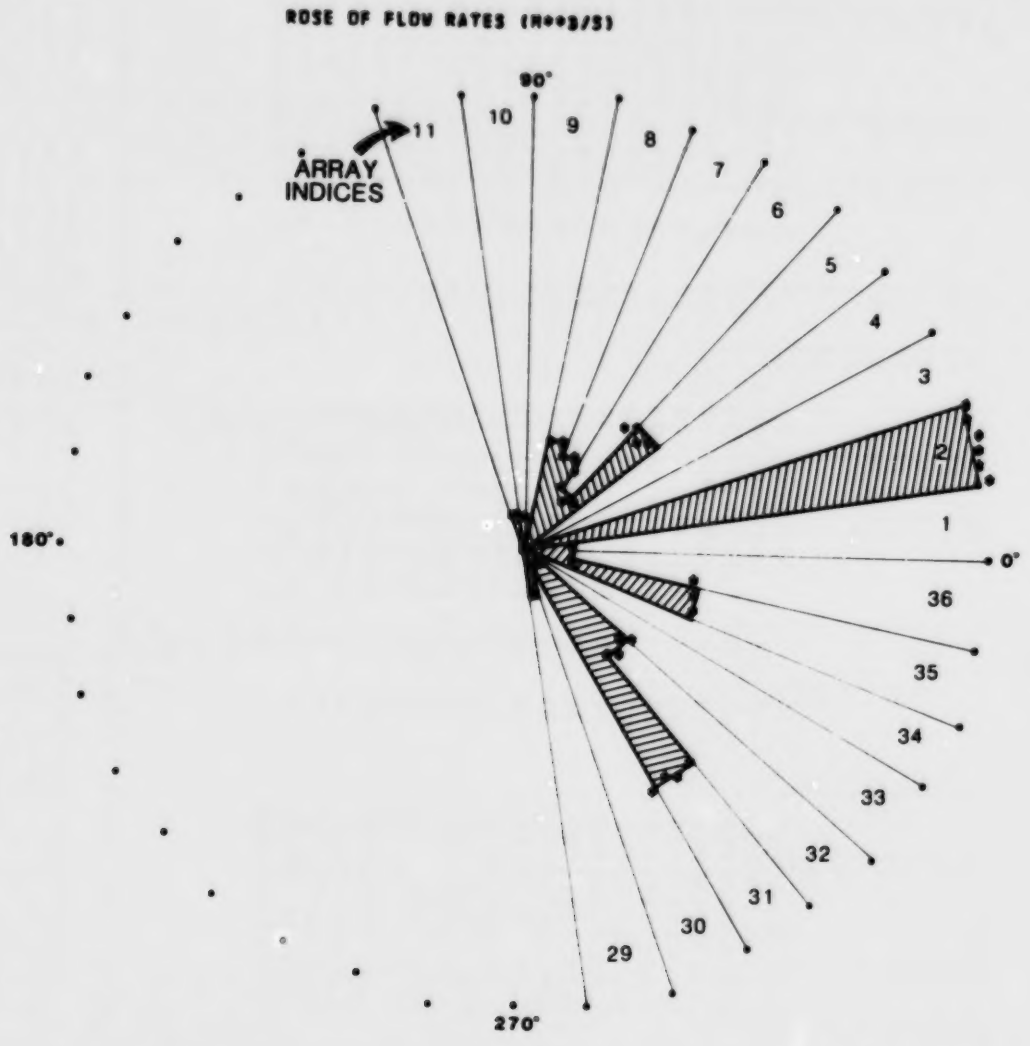


Figure F.3. Rose diagram of file SUMFLO completed by hand.

File FLOALL - unit 33

R990 SAMPLE RUN OF SDF PACK. : RECTANGULAR MODEL
FILE: FLOALL.U33

RESULTS OF FLOW CALCULATIONS, JAP= 1

HEAD AT NODAL POINTS

FORMAT(3(I IB(I) PHI(I)))

1	1	6.0000000E+00	2	1	6.0000000E+00	3	1	6.0000000E+00	4	5	2.0000000E+00
5	1	6.0000000E+00	6	5	2.0000000E+00	7	5	2.0000000E+00	8	1	6.0000000E+00
9	1	6.0000000E+00	10	5	2.0000000E+00	11	5	2.0000000E+00	12	1	6.0000000E+00
13	5	2.0000000E+00	14	5	2.0000000E+00	15	1	6.0000000E+00	16	5	2.0000000E+00
17	1	6.0000000E+00	18	0	5.7902527E+00	19	0	5.4793321E+00	20	0	3.6608947E+00
21	0	5.3714738E+00	22	0	5.3368393E+00	23	0	4.4447461E+00	24	0	4.8707362E+00
25	0	5.5560360E+00	26	0	5.0879165E+00	27	0	5.9231895E+00	28	0	5.8299749E+00
29	0	5.8683799E+00	30	0	5.7962460E+00	31	0	5.4560384E+00	32	0	5.9764863E+00
33	0	5.6939598E+00	34	0	4.7899931E+00	35	0	4.9667461E+00	36	0	3.3253596E+00
37	0	4.1488182E+00	38	0	4.8908659E+00	39	0	2.7432078E+00	40	0	3.9735566E+00
41	0	3.7675485E+00	42	0	2.0926704E+00	43	0	5.3978811E+00	44	0	5.3634930E+00
45	0	4.7307904E+00	46	0	4.8996332E+00	47	0	5.5464532E+00	48	0	5.9165804E+00
49	0	4.8320554E+00	50	0	5.4697596E+00	51	0	5.3127774E+00	52	0	2.0004003E+00
53	0	2.8055567E+00	54	0	2.6033181E+00	55	0	2.0823763E+00	56	0	2.4264833E+00
57	0	2.0188956E+00	58	0	2.4127811E+00	59	0	2.6120505E+00	60	0	3.1499411E+00
61	0	3.8389775E+00	62	0	3.3819204E+00	63	0	3.9270029E+00	64	0	3.9952934E+00
65	0	3.9911807E+00	66	0	4.9198695E+00	67	0	4.8789011E+00	68	0	4.7210369E+00
69	0	4.4793778E+00	70	0	5.4206777E+00	71	0	5.9459499E+00	72	0	5.9972814E+00
73	0	4.1225256E+00	74	0	2.2432608E+00	75	0	2.6516954E+00	76	0	5.3716782E+00
77	0	5.1269097E+00	78	0	4.5692202E+00	79	0	4.8622327E+00	80	0	4.8356400E+00
81	0	4.8429077E+00	82	0	5.7392669E+00	83	0	5.9063916E+00	84	0	3.9994167E+00

FLOW IN ELEMENTS, JAP= 1

NUMBER	WIDTH(M)	LENGTH(M)	VELOC (M/SEC)	FL.RATE(M**3/SEC)	REYNOLD NO.	DIRECTION	TRANSIT TIME
1	5.0000000E-06	1.8129744E+00	-2.3578586E-06	-1.1789293E-11	1.1765762E-08	1.6343680E+02	7.6873750E+05
2	5.0000000E-06	1.8563956E+00	-8.6036011E-06	-4.3018006E-11	4.2932141E-08	1.6007307E+02	2.1576961E+05
3	5.0000000E-06	1.9020807E+00	-4.5633917E-06	-2.2816958E-11	2.2771416E-08	1.6007307E+02	4.1681294E+05
4	5.0000000E-06	7.9373426E-01	-5.5752250E-06	-2.7876125E-11	2.7820484E-08	1.6007307E+02	1.4236811E+05
5	5.0000000E-06	9.0519125E-01	-5.6032684E-06	-2.8016342E-11	2.7960421E-08	1.6007307E+02	1.6154701E+05
6	5.0000000E-06	8.0085488E-02	-8.8119639E-06	-4.4059819E-11	4.3971876E-08	1.6007307E+02	9.0882679E+03

File TRANSIT - unit 41

R990 SAMPLE RUN OF SDF PACK. : RECTANGULAR MODEL
FILE : TRANSIT.U41

AUXILIARY DIRECTIONAL PARAMETERS FOR JAP= 1

DIRECTION	RDFR	DLM	DWTVM
1	1.88E-03	2.01E-01	8.35E-07
2	2.72E-01	7.47E-01	7.93E-06
3	8.83E-03	1.17E+00	2.52E-06
4	.00E+00	.00E+00	.00E+00
5	9.28E-02	5.50E-01	6.48E-06
6	3.97E-02	6.42E-01	3.99E-06
7	5.69E-02	7.61E-01	3.98E-06
8	6.62E-02	9.87E-01	4.42E-06
9	.00E+00	.00E+00	.00E+00
10	.00E+00	.00E+00	.00E+00
11	1.67E-02	9.31E-01	2.95E-06
12	.00E+00	.00E+00	.00E+00
13	.00E+00	.00E+00	.00E+00
14	.00E+00	.00E+00	.00E+00
15	.00E+00	.00E+00	.00E+00
16	.00E+00	.00E+00	.00E+00
17	.00E+00	.00E+00	.00E+00
18	.00E+00	.00E+00	.00E+00
19	.00E+00	.00E+00	.00E+00
20	.00E+00	.00E+00	.00E+00
21	.00E+00	.00E+00	.00E+00
22	.00E+00	.00E+00	.00E+00
23	.00E+00	.00E+00	.00E+00
24	2.77E-03	1.46E+00	1.23E-06
25	.00E+00	.00E+00	.00E+00
26	3.22E-03	6.93E-01	1.06E-06
27	.00E+00	.00E+00	.00E+00
28	.00E+00	.00E+00	.00E+00
29	3.16E-02	7.74E-01	4.83E-06
30	2.16E-02	1.11E+00	3.60E-06
31	1.61E-01	8.60E-01	3.72E-06
32	8.08E-02	1.25E+00	5.76E-06
33	1.01E-02	4.23E-01	4.46E-06
34	.00E+00	.00E+00	.00E+00
35	1.04E-01	1.01E+00	6.23E-06
36	2.93E-02	1.26E+00	4.45E-06

ARRAY IDIR
FORMAT: 10(J IDIR(J))

1	2	2	2	3	2	4	2	5	2	6	2	7	2	8	2	9	2	10	2
11	2	12	2	13	2	14	2	15	2	16	2	17	2	18	2	19	2	20	2
21	2	22	2	23	2	24	2	25	2	26	2	27	2	28	3	29	5	30	5
31	5	32	5	33	5	34	5	35	5	36	5	37	5	38	6	39	6	40	6
41	6	42	7	43	7	44	7	45	7	46	7	47	7	48	8	49	8	50	8
51	8	52	8	53	8	54	8	55	11	56	11	57	29	58	29	59	29	60	30
61	30	62	31	63	31	64	31	65	31	66	31	67	31	68	31	69	31	70	31
71	31	72	31	73	31	74	31	75	31	76	31	77	31	78	32	79	32	80	32
81	32	82	32	83	32	84	32	85	32	86	33	87	35	88	35	89	35	90	35
91	35	92	35	93	35	94	35	95	35	96	35	97	36	98	36	99	36	100	0

SIZE OF GENERATED NETWORK :
X-LENGTH= 10.0000 (M)
Y-LENGTH= 10.0000 (M)

PARTICLES MUST TRAVEL 100.00 (M) IN THE X DIRECTION

File TRANSIT - unit 41 (continued)

STATISTICS OF ALL PARTICLES FOR JAP= 1

NT	XTRAV(M)	YTRAV(M)	TRAV(M)	TIME(S)	NSTEP
1	1.00E+02	4.38E+00	1.44E+02	2.95E+07	169
2	1.00E+02	-3.38E+00	1.49E+02	3.12E+07	174
3	1.00E+02	4.87E+00	1.48E+02	2.99E+07	176
4	1.00E+02	4.91E+00	1.51E+02	3.17E+07	177
5	1.00E+02	-4.88E+00	1.45E+02	2.91E+07	173
6	1.00E+02	-1.39E+01	1.38E+02	2.68E+07	164
7	1.00E+02	-8.30E+00	1.51E+02	3.14E+07	178
8	1.00E+02	9.81E+00	1.40E+02	2.74E+07	171
9	1.00E+02	3.61E+00	1.53E+02	3.13E+07	181
10	1.00E+02	-1.69E+01	1.37E+02	2.74E+07	156
11	1.00E+02	-1.17E+01	1.52E+02	3.16E+07	179
12	1.00E+02	-1.95E+00	1.41E+02	2.77E+07	166
13	1.00E+02	8.32E-01	1.37E+02	2.59E+07	164
14	1.00E+02	4.92E+00	1.37E+02	2.57E+07	164
15	1.00E+02	2.38E+00	1.44E+02	2.90E+07	171
16	1.00E+02	1.46E+01	1.47E+02	2.93E+07	175
17	1.00E+02	-4.74E+00	1.47E+02	2.95E+07	168
18	1.00E+02	-9.15E+00	1.45E+02	2.89E+07	167
19	1.00E+02	-3.24E+00	1.42E+02	2.84E+07	167
20	1.00E+02	-2.31E+00	1.45E+02	2.90E+07	176
21	1.00E+02	6.34E+00	1.50E+02	3.03E+07	184
22	1.00E+02	-7.57E-01	1.43E+02	2.78E+07	171
23	1.00E+02	-6.46E+00	1.46E+02	2.97E+07	172
24	1.00E+02	-1.35E+00	1.46E+02	2.90E+07	174
25	1.00E+02	-2.66E+00	1.41E+02	2.77E+07	163
26	1.00E+02	-3.48E+00	1.45E+02	2.98E+07	170
27	1.00E+02	-4.82E+00	1.41E+02	2.81E+07	166
28	1.00E+02	-2.97E+00	1.38E+02	2.68E+07	164
29	1.00E+02	-1.61E+01	1.44E+02	2.90E+07	171
30	1.00E+02	5.19E+00	1.48E+02	2.94E+07	177
31	1.00E+02	4.20E-01	1.42E+02	2.78E+07	168
32	1.00E+02	2.95E+00	1.42E+02	2.78E+07	166
33	1.00E+02	4.25E+00	1.43E+02	2.77E+07	168
34	1.00E+02	-1.43E+01	1.41E+02	2.78E+07	165
35	1.00E+02	1.48E+00	1.42E+02	2.80E+07	170
36	1.00E+02	2.49E+00	1.33E+02	2.50E+07	158
37	1.00E+02	-1.34E+01	1.35E+02	2.61E+07	158
38	1.00E+02	-3.90E-01	1.41E+02	2.85E+07	172
39	1.00E+02	-1.21E+01	1.43E+02	2.88E+07	166
40	1.00E+02	-4.37E+00	1.42E+02	2.87E+07	163
41	1.00E+02	-4.58E+00	1.43E+02	2.91E+07	171
42	1.00E+02	-9.28E-01	1.46E+02	3.01E+07	173
43	1.00E+02	-1.20E+01	1.45E+02	3.09E+07	167
44	1.00E+02	-1.18E+01	1.39E+02	2.82E+07	162
45	1.00E+02	-5.55E+00	1.41E+02	2.84E+07	165
46	1.00E+02	-1.54E+01	1.47E+02	3.05E+07	168
47	1.00E+02	1.00E+01	1.43E+02	2.81E+07	173
48	1.00E+02	-8.98E+00	1.43E+02	2.82E+07	170
49	1.00E+02	-3.46E+00	1.49E+02	3.01E+07	174
50	1.00E+02	7.85E+00	1.40E+02	2.79E+07	175

BASIC DESCRIPTIVE STATISTICS OF ABS.TRAV.DIST.,TRAV(M)

NUMBER OF DATA POINTS	MEAN	STANDARD DEVIATION	MAXIMUM	MINIMUM
50	1.434E+02	4.390E+00	1.528E+02	1.328E+02

File TRANSIT - unit 41 (end)

FREQUENCY DIAGRAMS OF TRANSIT TIME (S)

BASIC DESCRIPTIVE STATISTICS OF TRANSIT TIME (S)

NUMBER OF DATA POINTS	MEAN	STANDARD DEVIATION	MAXIMUM	MINIMUM
50	2.872E+07	1.519E+06	3.173E+07	2.501E+07

MIDDLE INTERV	FREQ	CUM REL FREQ	RELATIVE FREQUENCY DIAGRAM	CUMULATIVE FREQUENCY DIAGRAM
2.52E+07	1	2	1*	1*
2.55E+07	0	2	1	1*
2.59E+07	2	6	1**	1***
2.62E+07	1	8	1*	1****
2.65E+07	0	8	1	1****
2.69E+07	2	12	1**	1******
2.72E+07	0	12	1	1******
2.75E+07	5	22	1*****	1**********
2.79E+07	6	34	1******	1******************
2.82E+07	4	42	1****	1**********************
2.85E+07	4	50	1****	1**************************
2.89E+07	5	60	1*****	1**********************************
2.92E+07	5	70	1*****	1**
2.95E+07	3	76	1***	1***
2.99E+07	2	80	1**	1**
3.02E+07	3	86	1***	1**
3.06E+07	1	88	1*	1***
3.09E+07	1	90	1*	1**
3.12E+07	3	96	1***	1**
3.16E+07	1	98	1*	1***
3.19E+07	1	100	1*	1**

********** //// END OF LIST ////

********** //// END OF LIST ////

F.2 A Rectangular Model With Abutting Fractures: Realization 992

All input data are the same as realization 990, except for the value of ITERM, which is 1 for the fracture sets 2, 3, and 4.

File INP1 — unit 1

```
992 SAMPLE RUN OF SDF PACK. : RECTANG. MODEL WITH ABUTMENT
   4     1  3.0 990.  25.     1     1     1                2   50 100.  0
-1.0 11.0 11.0 -1.0  0.0 10.0 10.0  0.0
   1         0.0  0.0     1  6.0
   0
   1         0.0 10.0     0
   0
   1        10.0 10.0     1  2.0
   0
   1        10.0  0.0     0
   0
      0.40     2     2     1     0
      0.60     2     2     1     1
      0.80     2     2     1     1
      1.00     2     2     1     1
       1.4        0.80       15.0       10.0      165.0       10.0  0.000005       0.0
       1.2        0.80       65.0       10.0      125.0       10.0  0.000005       0.0
       1.0        0.80      125.0       10.0       65.0       10.0  0.000005       0.0
       0.8        0.80      165.0       10.0       15.0       10.0  0.000005       0.0
 1 13 44
```

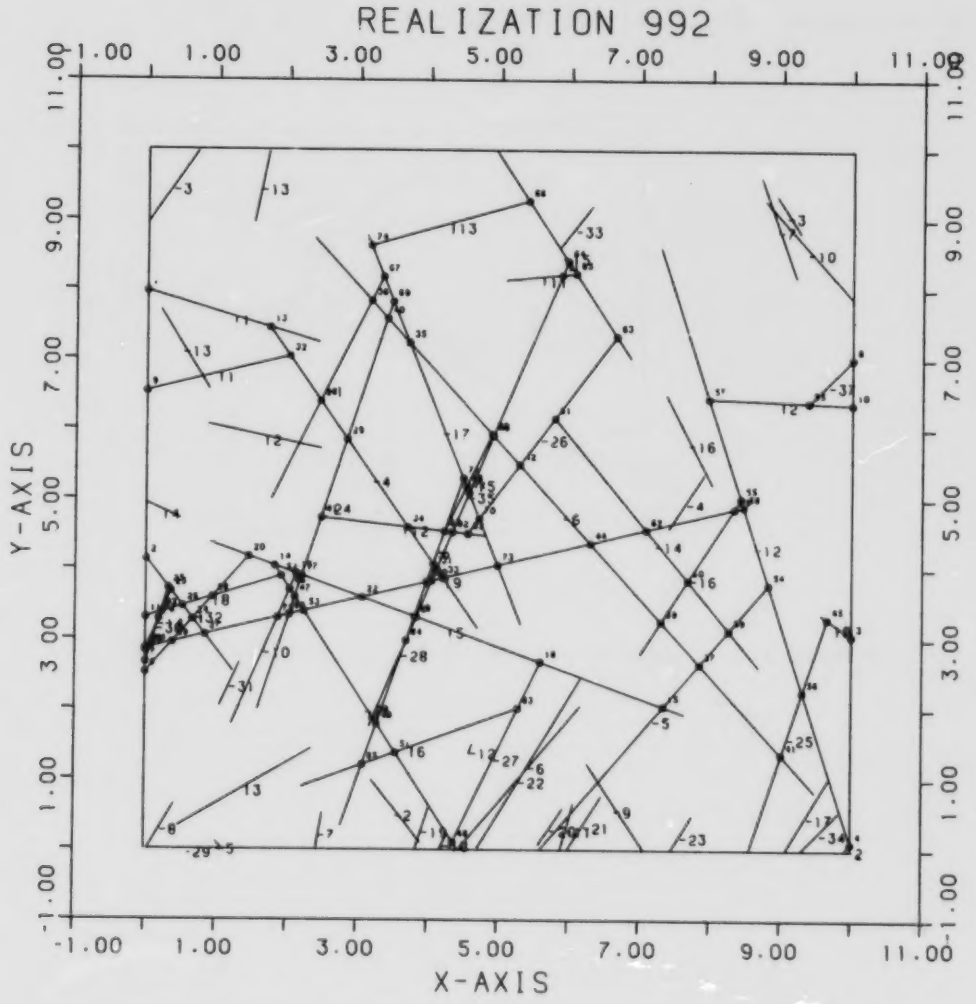


Figure F.4. Computer plot for realization 992; see Figure F.2 for meaning of the numbers.

F.3 A Circular Model: Realization 991

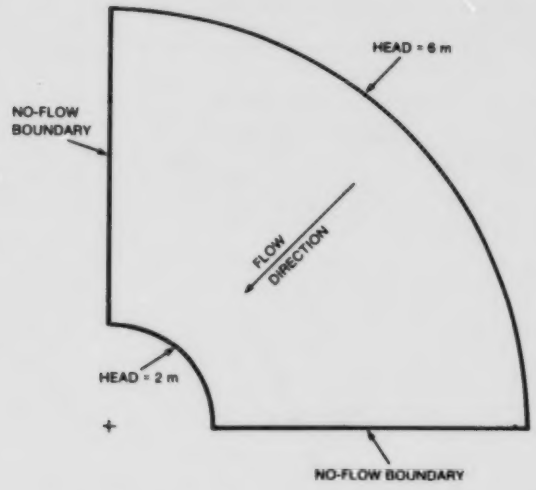


Figure F.5. Boundary conditions for realization 991.

File INP1 - unit 1

```
-991 SAMPLE RUN OF SDF PACK. : CIRCULAR MODEL
    4     2   6.0 991.  10.     2     1     1                    2
 -1.0 11.0 11.0 -1.0  0.0 10.0 10.0  0.0
    1        0.0  2.5    0
    0
    2 10.0  0.0 10.0    1  6.0
    1       10.0  0.0    0
    0
    2  2.5  2.5  0.0    1  2.0
      0.40    2    2    1    0
      0.60    2    2    1    0
      0.80    2    2    1    0
      1.00    2    2    1    0
       1.4       0.80       15.0       10.0      165.0       10.0  0.000005
       1.2       0.90       65.0       10.0      125.0       10.0  0.000005
       1.0       0.80      125.0       10.0       65.0       10.0  0.000005
       0.8       0.90      165.0       10.0       15.0       10.0  0.000005
```

File INP2 - unit 21

```
R991-1 FIRST SET OF APER. DATA FOR SAMPLE RUN ON CIRC. MODEL
    2    2    1    3       0.5 991.    4    0
     -13.0       0.8
     -12.0       0.8
     -11.0       0.8
     -10.0       0.8
```

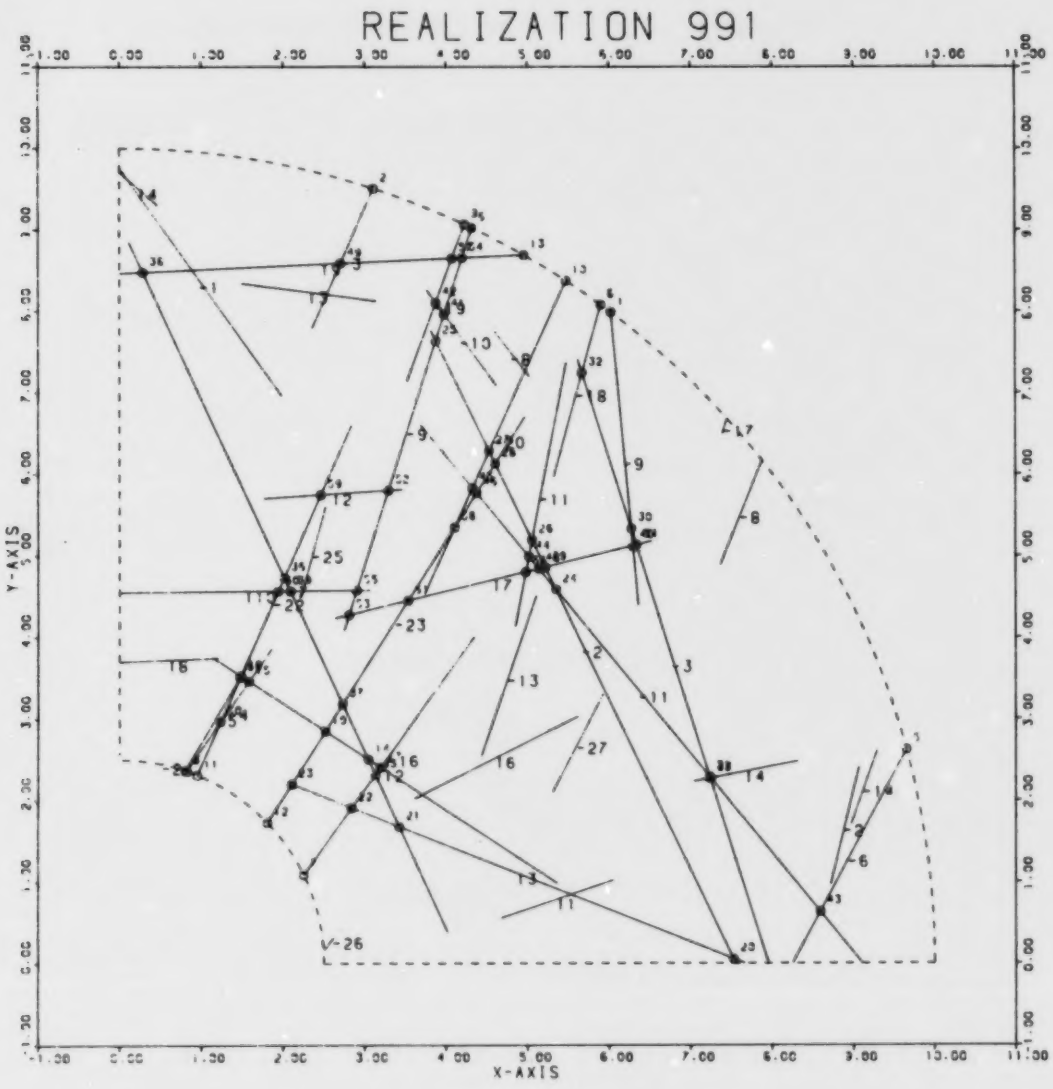


Figure F.6. Computer plot for realization 991; see Figure F.2 for meaning of the numbers.

Appendix G
Program Listings

APPENDIX G

Program Listings

G.1 Listing of the Program NETWRK

```
      PROGRAM NETWRK
C*****************************************************************
C
C  PROGRAM TO GENERATE A TWO-DIMENSIONAL NETWORK OF FINITE LINE SEGMENTS
C  AND TO COMPUTE SEVERAL STATISTICS ON THE GENERATED LINE NETWORK.
C  (WRITTEN BY A. ROULEAU, 1983; UPDATE BY A. ROULEAU, JAN. 1986)
C
C*****************************************************************
C
      DOUBLE PRECISION SEED,PI,RAD
      INTEGER THDSTR,APDSTR
C
C DIMENSION NSET = 4 , AND MAXFRA = 100
      DIMENSION THP(100,4)
      DIMENSION X1(100,4),Y1(100,4),X2(100,4),
     *Y2(100,4),TH(100,4),TANTH(100,4)
      INTEGER CUT1(100,4),CUT2(100,4)
C
C DIMENSION MAXFRA = 100
      DIMENSION XC(100),YC(100),TRACE(100),DEGRE(100),
     *DEGREP(100),APER(100)
C
C DIMENSION MAXINT=150; 2*MAXINT=300 (SEE DATA STATEMENT)
      DIMENSION XORD(150),YORD(150)
      INTEGER INC1(150,2),INC2(150,2),
     *IB(150)
      LOGICAL EFF(150)
C
C DIMENSION MAXELT = 200
      DIMENSION ELCR(200),ELORP(200),W(200),ELLEN(200)
      INTEGER NOD1(200),NOD2(200),ISET(200)
C
      COMMON /AREA1A/ XMINO,YMAXO,XMAXO,YMINO
      COMMON /AREA1B/ XMINI,YMAXI,XMAXI,YMINI,IGEOM
      COMMON /AREA2/ NSET,MAXFRA,MAXINT,MAXELT,NUMINT,NUMNPT,NUMELT,
     *NOINT,IPRT
      COMMON /AREA3A/ SEED,PI,RAD,COEF
      COMMON /AREA3B/ TITLE(20)
      COMMON /AREA4/ ATH(5),BTH(5),ATHP(5),BTHP(5),THDSTR(5),NFRAC(5)
      COMMON /AREA5/ DENS(5),ALE(5),ALE(5),LDSTR(5),ITERM(5)
      COMMON /AREA6/ AAP(5),BAP(5),APDSTR(5),DAAP,NAAP,IAAP,NAP,IAP
      COMMON /AREA7/ R(8),XB(8),YB(8),ISHAP(8),IBC(8)
C
      EQUIVALENCE (TH,TANTH),(DEGRE,APER),(TRACE,EFF,NOD1)
      EQUIVALENCE (IB,NOD2),(W,XC)
C
C DIMENSION MAXINT=150  (2*MAXINT=300)
      DATA IB/150*0/,INC2/300*0/
C
C ASSIGN VALUES TO SOME CONSTANTS
      MAXFRA=100
      MAXINT=150
      MAXELT=200
      PI=3.1415926536
      RAD=PI/180.
```

```
NETWRK


C
C  READ IN DATA
C
      READ(1,3) TITLE
    3 FORMAT(20A4)
      READ(1,5)NSET,ISP,COEF,SEED,PLTSIZ,IGEOM,NAP,NAAP,DAAP,IPRT
    5 FORMAT(2I5,3F5.0,3I5,F10.0,I5)
      READ(1,7) XMINO,YMAXO,XMAXO,YMINO,XMINI,YMAXI,XMAXI,YMINI
    7 FORMAT(8F5.0)
C
C  CHECK FOR MIXED UP RECTANGULAR BOUNDARIES
      IF(XMINI .GT. XMINO .AND. XMAXI .LT. XMAXO .AND. YMINI .GT. YMINO
     *.AND. YMAXI .LT. YMAXO) GO TO 19
      WRITE(2,8)
    8 FORMAT(///,'1** ERROR IN INPUT DATA: MIXED RECTANGULAR BOUNDARIES')
      STOP
C
   19   IF(IGEOM .EQ. 1) NBO=8
        IF(IGEOM .EQ. 2) NBO=6
      READ(1,22)(ISHAP(I),R(I),XB(I),YB(I),IBC(I),I=1,NBO)
   22 FORMAT(I5,3F5.0,I5)
      READ(1,25)(DENS(I),LDSTR(I),THDSTR(I),APDSTR(I),ITERM(I),I=1,NSET)
   25 FORMAT(F10.0,4I5)
      READ(1,29) (ALE(I),BLE(I),ATHP(I),BTHP(I),
     *ATH(I),BTH(I),AAP(I),BAP(I),I=1,NSET)
   29 FORMAT(8F10.0)
      WRITE(2,34) TITLE
34    FORMAT('1',//,1X,20A4,//,' FILE: SUMNET.U02')
C
C PRINT OUT BOUNDARY DATA
C
      WRITE(2,70)
   70 FORMAT(////,10X,'INPUT BOUNDARY DATA',///,' NUMBER SHAPE RADIUS ',
     *6X,'XB',9X,'YB',2X,'IBC',//)
      WRITE(2,71)(I,ISHAP(I),R(I),XB(I),YB(I),IBC(I),I=1,NBO)
   71 FORMAT(I5,I7,F8.2,2F10.3,I4)
C
C PRINT INPUT DATA FOR FRACTURE NETWORK
C
      WRITE(2,80)(I,I=1,NSET)
   80 FORMAT(////,10X,'INPUT DATA FOR FRACTURE NETWORK',///,' FRACTURE ',
     *'SET :',T25,5I10)
      WRITE(2,85)(DENS(I),I=1,NSET)
   85 FORMAT(/' DENSITY :',T25,5F10.3,//)
      WRITE(2,87) (ITERM(I),I=1,NSET)
   87 FORMAT(//,' TERMINATION MODE :',T25,5I10)
      WRITE(2,95)(LDSTR(I),I=1,NSET)
   95 FORMAT(/' DISTRIBUTION TYPE',//,'    FOR TRACE LENGTH :',
     *T25,5I10,//)
      WRITE(2,100)(THDSTR(I),I=1,NSET)
  100 FORMAT('    FOR ORIENTATION :',T25,5I10)
C
      WRITE(2,108) (ALE(I),I=1,NSET)
  108 FORMAT(/' DISTRIBUTION PARAMETERS',///,'  FOR TRACE LENGTH',//,10X,
     *'A :',T25,5F10.2)
      WRITE(2,110) (BLE(I), I=1,NSET)
  110 FORMAT(10X,'B :',T25,5F10.2)
```

```
      WRITE(2,112) (ATH(I),I=1,NSET)
  112 FORMAT(/,'  FOR ORIENTATION WITHIN SIMULATION PLANE',/,10X,
     *'A :',T25,5F10.2)
      WRITE(2,115) (BTH(I), I=1,NSET)
  115 FORMAT(10X,'B :',T25,5F10.2)
      WRITE(2,117) (ATHP(I),I=1,NSET)
  117 FORMAT(/,'  FOR ORIENTATION FROM SIMULATION PLANE',/,10X,
     *'A :',T25,5F10.2)
      WRITE(2,119) (BTHP(I),I=1,NSET)
  119 FORMAT(10X,'B :',T25,5F10.2)
C
C  GENERATE THE FRACTURE TRACES.
C
      CALL GENLIN(X1,Y1,X2,Y2,XC,YC,TRACE,DEGRE,DEGREP,
     *TH,THP,IB,XORD,YORD,INC1,CUT1,CUT2)
C
      IF(ISP-1) 130,125,126
C
C
C  SAMPLE THE LINE NETWORK FOR SPACING VALUES.
C
  125 CALL SPCNG1(TANTH,X1,Y1,X2,Y2)
      GO TO 130
  126 CALL SPCNG2(TANTH,X1,Y1,X2,Y2)
C
C
C  LOCATE ALL THE INTERSECTIONS.
C
  130 CALL INTERS(TANTH,X1,Y1,X2,Y2,INC1,INC2,XORD,YORD,IB)
C
C
C  FIND OUT THE EFFECTIVENESS OF EVERY INTERSECTION.
C
      CALL EFFINT(X1,INC1,INC2,EFF,XORD,YORD,IB)
C
      IF(PLTSIZ .LT. 0.0) GO TO 140
C
C
C  MAKE A PLOT OF THE FRACTURE NETWORK.
C
      CALL PLOTFR(PLTSIZ,TANTH,X1,Y1,X2,Y2,XORD,YORD)
C
C
C  DEFINE THE ELEMENTS.
C
  140 CALL ELTDEF(X1,Y1,X2,Y2,INC1,INC2,XORD,YORD,APER,CUT1,CUT2,
     *TH,THP,ELOR,ELORP,ELLEN,W,NOD1,NOD2,ISET)
C
      STOP
      END
```

```
C ***********************************************************************
C
C  THE SUBROUTINE GENLIN GENERATES THE LINE TRACES FOR ALL THE
C  FRACTURE SETS, ONE SET AFTER THE OTHER. GENLIN CALLS THE SUBROUTINES
C  RECODR AND CIRCDR.
C
      SUBROUTINE GENLIN(X1,Y1,X2,Y2,XC,YC,TRACE,DEGRE,DEGREP,
     *TH,THP,IB,XORD,YORD,INC1,CUT1,CUT2)
C
      DOUBLE PRECISION SEED,PI,RAD
      INTEGER ESTNFR,THDSTR
      COMMON /AREA1A/ XMINO,YMAXO,XMAXO,YMINO
      COMMON /AREA1B/ XMINI,YMAXI,XMAXI,YMINI,IGEOM
      COMMON /AREA2/ NSET,MAXFRA,MAXINT,MAXELT,NUMINT,NUMNPT,NUMELT,
     *NOINT,IPRT
      COMMON /AREA3A/ SEED,PI,RAD,COEF
      COMMON /AREA4/ ATH(5),BTH(5),ATHP(5),BTHP(5),THDSTR(5),
     *NFRAC(5)
      COMMON /AREA5/ DENS(5),ALE(5),BLE(5),LDSTR(5),ITERM(5)
      COMMON /AREA7/ R(8),XB(8),YB(8),ISHAP(8),IBC(8)
      COMMON /AREA9/ DESLEN,SUMLE,ESTNFR
      COMMON /AREA15/ JSET,JFRA,JO,NBN,IBL(2)
      COMMON /AREA16/ LINT
C
      INTEGER IB(MAXINT),INC1(MAXINT,2)
      INTEGER CUT1(MAXFRA,NSET),CUT2(MAXFRA,NSET)
      DIMENSION TH(MAXFRA,NSET),THP(MAXFRA,NSET),
     *X1(MAXFRA,NSET),Y1(MAXFRA,NSET),X2(MAXFRA,NSET),Y2(MAXFRA,NSET)
      DIMENSION XC(MAXFRA),YC(MAXFRA),TRACE(MAXFRA),DEGRE(MAXFRA),
     *DEGREP(MAXFRA)
      DIMENSION XORD(MAXINT),YORD(MAXINT)
C
C
C COMPUTE SURFACE AREA OF MODEL
      IF(IGEOM .EQ. 1) THEN
        AREA=(XMAXI-XMINI)*(YMAXI-YMINI)
      ELSE
        RO=R(3)
        RI=R(6)
        AREA=PI*(RO-RI)*(RO+RI)/4.
      END IF
C
      LINT=0
C
      DO 220 JSET=1,NSET
C
C CALCULATE DESIRED TOTAL TRACE LENGTH (DESLEN) AND ESTIMATED NUMBER OF
C TRACES (ESTNFR) FOR THE SET.
      I=JSET
      A1=ATHP(I)*RAD
      DENS(I)=DENS(I)*SIN(A1)
      DESLEN = DENS(I) * AREA
      IF(LDSTR(I) .EQ. 2) GO TO 16
      AMEAN=ALE(I)
      GO TO 18
```

```
   16 AMEAN = EXP(ALE(I)+ BLE(I)*BLE(I)*0.5)
   14 ESTNFR = (DESLEN/AMEAN) * COEF
      IF(ESTNFR .LT. MAXFRA) GO TO 24
      WRITE(2,20) ESTNFR,I,MAXFRA
   20 FORMAT('1**** ERROR CONCERNING MAXFRA **********',///,
     *' ESTNFR = ',I4,' FOR SET ',I4,'. THIS IS MORE THAN THE MAXIMUM'
     */,'FOR WHICH MEMORY SPACE IS AVAILABLE, I.E. ',I4,' (MAXFRA)'
     *,//' DECREASE COEF OR INCREASE THE VALUE OF MAXFRA IN THE PROGRAM')
      STOP
   24 IF(IPRT .EQ. 1)GO TO 26
      WRITE(2,25) I
   25 FORMAT(////,10X,'FRACTURES  OF  SET',I4,/4X,'NUMBER',//,
     *' GNRTD ACTUAL', 4X,'LENGTH',5X,'ANGLE',6X,'X1',8X,'Y1',8X,'X2',
     *3X,'Y2',/)
C
C    GENERATE FRACTURE CENTERS.
C
   26  CALL GGUBS(SEED,ESTNFR,XC)
       CALL GGUBS(SEED,ESTNFR,YC)
C
C    GENERATE  FRACTURE TRACE LENGTHS .
C
       IF(LDSTR(I) .EQ. 2) GO TO 30
       DO 27 J=1,ESTNFR
   27  TRACE(J) = ALE(I)
       GO TO 45
   30  CALL GGNLG(SEED,ESTNFR,ALE(I),BLE(I),TRACE)
C
C    GENERATE FRACTURE ORIENTATIONS .
C
   45  IF(THDSTR(I) .EQ. 2) GO TO 55
       DO 52 J=1,ESTNFR
       THP(J,I) = A1
   52  DEGRE(J)= ATH(I)
       GO TO 60
   55  CALL GGNML(SEED,ESTNFR,DEGRE)
       CALL GGNML(SEED,ESTNFR,DEGREP)
       DO 58 J=1,ESTNFR
       THP(J,I)= (DEGREP(J)*BTHP(I) + ATHP(I))*RAD
   58  DEGRE(J)= DEGRE(J)*BTH(I) + ATH(I)
C
C    COMPUTE COORDINATES OF EXTREMITIES OF ALL FRACTURES IN SET I.
C
   60  DO 70  J = 1,ESTNFR
       THL=DEGRE(J)
C
C   TO AVOID SINGULARITY AND EVENTUAL DIVISION BY 0., MAKE SURE THL IS
C   NOT EXACTLY -90.0,0.0,90.0,180.0 OR 270.0.
         IF((THL .GE. 0.50 .AND. THL .LE. 179.50) .OR. (THL .LE. -0.50)
     *   .OR. (THL .GE. 180.50)) GO TO 62
         IF((THL .LE. 0.5 .AND. THL .GE. 0.0) .OR.
     *     (THL .LE. 180.5 .AND. THL .GE. 180.0)) THEN
           THL=0.5
         ELSE IF((THL .LE. 0.0 .AND. THL .GE. -0.5) .OR.
     *     (THL .LE. 180.0 .AND. THL .GE. 179.5)) THEN
           THL=-0.5
         END IF
```

```
      GO TO 63
   62 IF((THL .LE. 89.5 .AND. THL .GE. -89.5) .OR.
     *    (THL .GE. 90.5 .AND. THL .LE. 269.5)) GO TO 63
      IF((THL .GE. 89.5 .AND. THL .LE. 90.0) .OR.
     *    (THL .GE. 269.5 .AND. THL .LE. 270.0)) THEN
         THL=89.5
      ELSE IF((THL .GE. 90.0 .AND. THL .LE. 90.5) .OR.
     *       (THL .GE. -90.0 .AND. THL .LE. -89.5)) THEN
         THL=90.5
      END IF
   63 THL=THL*RAD
      XCJ = XC(J)*XMAXO + (1-XC(J))*XMINO
      YCJ = YC(J)*YMAXO + (1-YC(J))*YMINO
      TRL=TRACE(J)
      DY = TRL*SIN(THL) * 0.5
      DX = TRL*COS(THL) * 0.5
      X1L = XCJ + DX
      Y1L = YCJ + DY
      X2L = XCJ - DX
      Y2L = YCJ - DY
C
C  TO SIMPLIFY COMPUTATIONS, MAKE SURE Y2 IS LT Y1.
C
      IF(Y2L .LT. Y1L) GO TO 65
      X3= X1L
      Y3= Y1L
      X1L = X2L
      Y1L = Y2L
      X2L = X3
      Y2L = Y3
C
C ARE THE FRACTURES OF THIS SET ABUTTING ON FRACTURES OF OTHER SETS ?
   65       WRITE(2,555)I,ITERM(I)
  555       FORMAT(' BEFORE TERM: I, ITERM(I):',2I5)
      IF (ITERM(I) .EQ. 1) THEN
         CALL TERM(JSET,THL,X1L,Y1L,X2L,Y2L,TH,X1,Y1,X2,Y2)
      END IF
C
C  CONVERT THE VALUE OF THL FROM ANGLE TO TANGENT AND ENTER VALUES IN
C  CORRESPONDING ARRAY.
C
      TH(J,I)=TAN(THL)
      X1(J,I)=X1L
      Y1(J,I)=Y1L
      X2(J,I)=X2L
      Y2(J,I)=Y2L
   70 CONTINUE
C
C   CHECK FOR ANY OF THE EXTREMITIES GOING OUTSIDE THE MODEL BOUNDARIES
C   AND MAKE ADJUSTMENT FOR THE TRACE LENGTH AND FOR THE COORDINATES
C    OF THE CHOPPED OUT EXTREMITY.
C
      NBN=0
      SUMLE=0.
      JO=0
      IF(IGEOM .EQ. 2)GO TO 80
      CALL RECBDR(X1,Y1,X2,Y2,TRACE,TH,DEGRE,IB,XORD,YORD,INC1,
```

```
     *CUT1,CUT2)
      GO TO 90
   80 CALL CIRBDR(X1,Y1,X2,Y2,TRACE,TH,DEGRE,IB,XORD,YORD,INC1,
     *CUT1,CUT2)
C
C  PRINT SUMMARY OF FRACTURE SET I.
C
   90 IF(SUMLE .GT. DESLEN) GO TO 180
      WRITE(2,175) DESLEN,SUMLE,ESTNFR
  175 FORMAT(///,'1**** ERROR IN SUBROUTINE GENLIN ****',///,
     *' ESTIMATED NUMBER OF FRACTURES IS TOO SMALL',///,
     *' INCREASE THE VALUE OF COEF.',///,
     *10X,'DESLEN = ',F10.4,' SUMLE = ',F10.4,' ESTNFR = ',I4)
      STOP
  180 ACTLEN = SUMLE
      NFRAC(I) = JFRA
      WRITE(2,214) I, DENS(I),AMEAN,AREA,DESLEN,ACTLEN,ESTNFR,NFRAC(I),
     *JO,NBN
  214 FORMAT(////////,20X,'SUMMARY OF FRACTURE SET',I4,
     *       //,2X,'DENSITY OF FRACTURING ',F10.4,' L/L**2',/
     *         ,2X,'THEORETICAL MEAN TRACE LENGTH',F10.4,/
     *         ,2X,'SURFACE AREA OF THE MODEL',F10.4,/
     *         ,2X,'DESIRED TOTAL LENGTH FOR THE SET',F10.4,/
     *         ,2X,'ACTUAL TOTAL LENGTH FOR THE SET ',F10.4,/
     *         ,2X,'ESTIMATED NUMBER OF FRACTURES',I4,/
     *         ,2X,'ACTUAL NUMBER OF FRACTURES GENERATED ',I4,/
     *   ,2X,'NUMBER OF FRACTURES TOTALLY IN THE BUFFER MARGIN ',I4,/
     *    ,2X,'NUMBER OF INTERSECTIONS WITH A BOUNDARY',I4)
  220 CONTINUE
C
      RETURN
      END
```

TERM

```
      SUBROUTINE TERM(JSET,THL,X1L,Y1L,X2L,Y2L,TANTH,X1,Y1,X2,Y2)
C
C ***************************************************************
C
C THIS SUBROUTINE ENSURES THAT THE EXTREMITY 1 (HIGHER Y-VALUE) ALWAYS
C TERMINATES ON ANOTHER FRACTURE, IF THE FRACTURE DOES INTERSECT
C ANOTHER FRACTURE.
C
C ***************************************************************
C
      COMMON /AREA2/ NSET,MAXFRA,MAXINT,MAXELT,NUMINT,NUMNPT,NUMELT,
     *NOINT,IPRT
      COMMON /AREA4/ ATH(5),BTH(5),ATHP(5),BTHP(5),THDSTR(5),NFRAC(5)
      COMMON /AREA5/ DENS(5),ALE(5),BLE(5),LDSTR(5),ITERM(5)
      DIMENSION TANTH(MAXFRA,NSET),X1(MAXFRA,NSET),Y1(MAXFRA,NSET),
     *X2(MAXFRA,NSET),Y2(MAXFRA,NSET)
C
        WRITE(2,5)JSET,X1L,Y1L,X2L,Y2L
  5     FORMAT(' OLD: JSET, X1L,Y1L,X2L,Y2L:',I5,4F10.2)
      IF (JSET .EQ. 1) THEN
        WRITE(2,10)
 10     FORMAT(///,' **** WARNING ****',//,
     *' FRACTURE SET 1 CANNOT HAVE ABUTTING FRACTURES, ',//,
     *' I.E. ITERM(1)=1 IS NOT PERMITTED.',////)
            RETURN
        END IF
C FIND ALL INTERSECTIONS OF THE FRACTURE CONSIDERED WITH THE
C FRACTURES OF THE SETS ALREADY GENERATED. THIS SECTION IS SIMILAR
C TO A PORTION OF SUBROUTINE INTERS.
C
      TAN1=1/TAN(THL)
      YMAX=0.
      DO 110 I=1,JSET-1
         NFRACI=NFRAC(I)
        DO 100 J=1,NFRACI
          TAN2=1/TANTH(J,I)
C COMPUTE INTERSECTION ORDINATE YI AND CHECK FOR YI INSIDE THE RANGE
C FOR BOTH FRACTURES.
          IF ((TAN1-TAN2) .EQ. 0.0) TAN1=TAN2+0.000001
          YI=((Y1L*TAN1)-(Y1(J,I)*TAN2)-X1L+X1(J,I)) / (TAN1-TAN2)
          IF (YI .LT. Y2L .OR. YI .GT. Y1L) GO TO 100
          IF (YI .LT. Y2(J,I) .OR. YI .GT. Y1(J,I)) GO TO 100
C CHECK FOR LARGEST Y-VALUE OF INTERSECTION
        IF (YI .GE. YMAX) YMAX=YI
C
 100   CONTINUE
 110  CONTINUE
      IF(YMAX .LE. 0.) RETURN
C
C CUT EXTREMITY 1
      XMAX=(YMAX-Y1L)*TAN1 + X1L
      DY=Y1L-YMAX
      DX=X1L-XMAX
      Y1L=YMAX+0.00001
      X1L=XMAX+0.00001*TAN1
```

```
 TERM


C
C ATTACH IT TO EXTREMITY 2
      X2L=X2L-DX
      Y2L=Y2L-DY
        WRITE(2,555)X1L,Y1L,X2L,Y2L
 555    FORMAT(' NEW:',22X,4F10.2)
C
      RETURN
      END
```

```
C ***********************************************************************
C
C  THE SUBROUTINE RECBDR, FOR RECTANGULAR BOUNDARIES, CHECKS FOR ANY LINE
C  EXTREMITY GOING OUTSIDE THE MODEL BOUNDARIES AND MAKES ADJUSTMENT FOR
C  TRACE LENGTH AND FOR THE COORDINATES OF THE CHOPPED OUT EXTREMITY(IES)
C  RECBDR CALLS THE SUBROUTINES CHOPXY AND BNODE.
C
      SUBROUTINE RECBDR(X1,Y1,X2,Y2,TRACE,TH,DEGRE,IB,XORD,YORD,INC1,
     *CUT1,CUT2)
      INTEGER ESTNFR,CUTL(2)
      LOGICAL ELIM
      COMMON /AREA1B/ XMINI,YMAXI,XMAXI,YMINI,IGEOM
      COMMON /AREA2/ NSET,MAXFRA,MAXINT,MAXELT,NUMINT,NUMNPT,NUMELT,
     *NOINT,IPRT
      COMMON /AREA7/ R(8),XB(8),YB(8),ISHAP(8),IBC(8)
      COMMON /AREA9/ DESLEN,SUMLE,ESTNFR
      COMMON /AREA12/ X1L,Y1L,X2L,Y2L,TRL,THL,CUTL
      COMMON /AREA15/ JSET,JFRA,JC,NBN,IBL(2)
      COMMON /AREA16/ LINT
C
      INTEGER IB(MAXINT),INC1(MAXINT,2)
      INTEGER CUT1(MAXFRA,NSET),CUT2(MAXFRA,NSET)
      DIMENSION X1(MAXFRA,NSET),Y1(MAXFRA,NSET),X2(MAXFRA,NSET),
     *Y2(MAXFRA,NSET),TH(MAXFRA,NSET)
      DIMENSION DEGRE(MAXFRA),TRACE(MAXFRA)
      DIMENSION XORD(MAXINT),YORD(MAXINT)
      EQUIVALENCE (LINT,L),(JSET,I),(JFRA,J)
C
      DO 170 JFRA=1,ESTNFR
C
      ELIM=.FALSE.
      CUTL(1)=0
      CUTL(2)=0
      X1L=X1(J,I)
      Y1L=Y1(J,I)
      X2L=X2(J,I)
      Y2L=Y2(J,I)
      TRL=TRACE(J)
      THL=TH(J,I)
      IF(Y2L .GE. YMAXI .OR. Y1L .LE. YMINI)GO TO 180
C
C CASES OF INTERSECTION WITH A Y-BOUNDARY
      IF(Y1L .LE. YMAXI)GO TO 90
C CASE OF INTERSECTION WITH UPPER BOUNDARY
      CALL CHOPXY(1,2,YMAXI)
C
      IF(ISHAP(4) .EQ. 0) THEN
        IBL(1)=3
      ELSE
        IF(X1L .LE. XB(4)) THEN
          IBL(1)=3
        ELSE
          IBL(1)=4
        END IF
      END IF
```

```
C
   80 IF(Y2L .GE. YMINI)GO TO 85
C CASE OF INTERSECTION WITH LOWER BOUNDARY
      CALL CHOPXY(2,2,YMINI)
      IF(ISHAP(8) .EQ. 0) THEN
        IBL(2)=7
      ELSE
        IF(X2L .LE. XB(8)) THEN
          IBL(2)=8
        ELSE
          IBL(2)=7
        END IF
      END IF
C
   85 IF(THL .LE. 0.0)GO TO 105
C
C CASE X2L .LT. X1L (I.E. THL IS POSITIVE)
      IF(X2L .GE. XMAXI .OR. X1L .LE. XMINI)GO TO 160
      IF(X1L .LE. XMAXI)GO TO 100
C CASE OF INTERSECTION WITH RIGHT BOUNDARY
      CALL CHOPXY(1,1,XMAXI)
      IF(ISHAP(6) .EQ. 0) THEN
        IBL(1)=5
      ELSE
        IF(Y1L .LE. YB(6)) THEN
          IBL(1)=6
        ELSE
          IBL(1)=5
        END IF
      END IF
C
  100 IF(X2L .GE. XMINI)GO TO 165
C CASE OF INTERSECTION WITH LEFT BOUNDARY
      CALL CHOPXY(2,1,XMINI)
      IF(ISHAP(2) .EQ. 0) THEN
        IBL(2)=1
      ELSE
        IF(Y2L .LE. YB(2)) THEN
          IBL(2)=1
        ELSE
          IBL(2)=2
        END IF
      END IF
C
      GO TO 165
C
C CASE X2L .GT. X1L (I.E. THL IS NEGATIVE)
  105 IF(X2L .LE. XMINI .OR. X1L .GE. XMAXI)GO TO 160
      IF(X1L .GE. XMINI)GO TO 115
C CASE OF INTERSECTION WITH LEFT BOUNDARY
      CALL CHOPXY(1,1,XMINI)
      IF(ISHAP(2) .EQ. 0) THEN
        IBL(1)=1
      ELSE
        IF(Y1L .LE. YB(2)) THEN
          IBL(1)=1
        ELSE
```

```
            IBL(1)=2
          END IF
        END IF
C
  115 IF(X2L .LE. XMAXI)GO TO 165
C CASE OF INTERSECTION WITH RIGHT BOUNDARY
        CALL CHOPXY(2,1,XMAXI)
        IF(ISHAP(6) .EQ. 0) THEN
          IBL(2)=5
        ELSE
          IF(Y2L .LE. YB(L)) THEN
            IBL(2)=6
          ELSE
            IBL(2)=5
          END IF
        END IF
C
        GO TO 165
C
  160 ELIM=.TRUE.
        X1L=999.
        JD=JD+1
  165 IF(IPRT .EQ. 1)GO TO 167
        WRITE(2,166)J,TRL,DEGRE(J),X1L,Y1L,X2L,Y2L
  166 FORMAT(I12,2X,6F10.3)
  167 X1(J,I)=X1L
        IF(ELIM) GO TO 170
C
C REENTER VALUES IN CORRESPONDING ARRAYS.
        SUMLE=SUMLE+TRL
        Y1(J,I)=Y1L
        X2(J,I)=X2L
        Y2(J,I)=Y2L
        TRACE(J)=TRL
C
C DEFINE BOUNDARY NODE(S)
        DO 168 MEND=1,2
        IF(CUTL(MEND) .EQ. 1) THEN
          CALL ANODE(MEND,IB,XORD,YORD,INC1)
        END IF
  168 CONTINUE
C
        CUT1(J,I)=CUTL(1)
        CUT2(J,I)=CUTL(2)
        IF(SUMLE .GE. DESLEN)GO TO 200
  170 CONTINUE
  200 RETURN
        END
```

```
C*********************************************************************
C
C THE SUBROUTINE CIRBDR, FOR CIRCULAR BOUNDARIES, CHECKS FOR ANY LINE
C EXTREMITY GOING OUTSIDE THE MODEL BOUNDARIES AND MAKES ADJUSTMENT
C FOR TRACE LENGTH AND FOR THE COORDINATES OF THE CHOPPED CUT
C EXTREMITY(IES). CIRBDR CALLS THE SUBROUTINES CIRCL2, CHOPXY, CHOPC
C AND BNODE.
C
C                  LOCAL VARIABLES (IN CIRBDR AND CIRCL2)
C
C CIRCL1 : LOGICAL STATEMENT FUNCTION THAT TAKES VALUE .TRUE. IF A GIVEN
C          POINT IS OUTSIDE A GIVEN CIRCLE.
C INTC1 & INTC2 :
C          LOGICAL VARIABLES THAT TAKE VALUE .TRUE. IF A GIVEN LINE
C          INTERSECTS A GIVEN CIRCLE (TWO POSSIBLE INTERSECTIONS).
C OUTC1 & OUTC2 :
C          LOGICAL VARIABLES WHOSE VALUE IS ASSIGNED BY THE STATEMENT
C          FUNCTION CIRCL1, FOR EXTREMITIES 1 & 2 RESPECTIVELY OF A
C          GIVEN LINE.
C SEPAR :  LOGICAL VARIABLE THAT TAKES VALUE .TRUE. IF A LINE IS CUT IN
C          TWO SEGMENTS INSIDE THE MODEL ITSELF BY THE INNER CIRCLE.
C XCENT & YCENT :
C          COORDINATES OF THE CENTER OF THE CIRCLES FORMING A BOUNDARY.
C XC1, YC1 & XC2, YC2 :
C          COORDINATES OF THE INTERSECTION(S) OF A LINE AND A CIRCLE
C          (POSSIBILITY OF TWO INTERSECTIONS).
C
      SUBROUTINE CIRBDR(X1,Y1,X2,Y2,TRACE,TH,DEGRE,IB,XORD,YORD,INC1,
     *CUT1,CUT2)
      INTEGER ESTNFR,CUTL(2),CUTEMP
      LOGICAL OUTC1,OUTC2,INTC1,INTC2,CIRCL1,SEPAR,ELIM
      COMMON /AREA1B/ XMINI,YMAXI,XMAXI,YMINI,IGEOM
      COMMON /AREA2/ NSET,MAXFRA,MAXINT,MAXELT,NUMINT,NUMNPT,NUMELT,
     *NOINT,IPRT
      COMMON /AREA7/ R(8),XB(8),YB(8),ISHAP(8),IBC(8)
      COMMON /AREA9/ DESLEN,SUMLE,ESTNFR
      COMMON /AREA12/ X1L,Y1L,X2L,Y2L,TRL,THL,CUTL
      COMMON /AREA14/ XCENT,YCENT
      COMMON /AREA15/ JSET,JFRA,JO,NBN,IBL(2)
      COMMON /AREA16/ LINT
C
      INTEGER IB(MAXINT),INC1(MAXINT,2)
      INTEGER CUT1(MAXFRA,NSET),CUT2(MAXFRA,NSET)
      DIMENSION X1(MAXFRA,NSET),Y1(MAXFRA,NSET),X2(MAXFRA,NSET),
     *Y2(MAXFRA,NSET),TH(MAXFRA,NSET)
      DIMENSION DEGRE(MAXFRA),TRACE(MAXFRA)
      DIMENSION XORD(MAXINT),YORD(MAXINT)
      EQUIVALENCE (LINT,L),(JSET,I),(JFRA,JJ)
C
C CIRCL1 IS A STATEMENT FUNCTION THAT FINDS OUT IF A POINT IS OUTSIDE
C A GIVEN CIRCLE OR NOT.
      CIRCL1(X4,Y4,R2)=((X4-XMINI)**2 + (Y4-YMINI)**2) .GE. R2
C
C INITIALIZE OR SET THE VALUE OF SOME VARIABLES.
      XCENT=XMINI
```

```
 CIRBD#

      YCENT=YMINI
      J=0
      JJ=0
      RI2=R(6)**2
      RO2=R(3)**2
C
   15 SEPAR=.FALSE.
      J=J+1
      JJ=JJ+1
      ELIM=.FALSE.
      CUTL(1)=0
      CUTL(2)=0
      X1L=X1(JJ,I)
      Y1L=Y1(JJ,I)
      X2L=X2(JJ,I)
      Y2L=Y2(JJ,I)
      TRL=TRACE(JJ)
      THL=TH(JJ,I)
C
C FIND OUT IF POINTS ARE OUTSIDE OUTER CIRCLE
      OUTC1=CIRCL1(X1L,Y1L,RO2)
      OUTC2=CIRCL1(X2L,Y2L,RO2)
C CASE BOTH POINTS ARE INSIDE OUTER CIRCLE
      IF(.NOT. OUTC1 .AND. .NOT. OUTC2)GO TO 30
C
C LOCATE INTERSECTION(S) WITH OUTER CIRCLE
      CALL CIRCL2(RO2,INTC1,INTC2,XC1,YC1,XC2,YC2)
      IF(OUTC1 .AND. OUTC2) THEN
C CASE BOTH POINTS ARE OUTSIDE OUTER CIRCLE
        IF(.NOT. INTC1 .AND. .NOT. INTC2) THEN
          GO TO 160
        ELSE
          CALL CHOPC(1,XC1,YC1)
          IBL(1)=3
          CALL CHOPC(2,XC2,YC2)
          IBL(2)=3
        END IF
      ELSE
C CASES ONE POINT IN AND ONE POINT OUT OUTSIDE CIRCLE
        IF(OUTC1) THEN
          CALL CHOPC(1,XC1,YC1)
          IBL(1)=3
        ELSE
          CALL CHOPC(2,XC2,YC2)
          IBL(2)=3
        END IF
      END IF
C
C CHECK FOR INTERSECTION WITH LOWER BOUNDARY
C
C CASE OF LINE TOTALLY IN BOTTOM MARGIN
   30 IF(Y1L .LE. YMINI)GO TO 160
      IF(Y2L .LE. YMINI) THEN
C CASE OF LINE OVERLAPPING LOWER BOUNDARY
        CALL CHOPXY(2,2,YMINI)
        IF(ISHAP(5) .EQ. 0) THEN
          IBL(2)=4
```

```
          ELSE
            IF(X2L .LE. XB(5)) THEN
              IBL(2)=5
            ELSE
              IBL(2)=4
            END IF
          END IF
        END IF
C
C CHECK FOR INTERSECTION WITH LEFT BOUNDARY
C
C CASE OF LINE TOTALLY IN LEFT MARGIN
      IF(X1L .LE. XMINI .AND. X2L .LE. XMINI)GO TO 160
C CASE OF LINE TOTALLY TO THE RIGHT OF LEFT BOUNDARY
      IF(X1L .GE. XMINI .AND. X2L .GE. XMINI)GO TO 40
C CASES OF LINE OVERLAPPING LEFT BOUNDARY
      IF(X1L .LE. XMINI) THEN
        CALL CHOPXY(1,1,XMINI)
        IF(ISHAP(2) .EQ. 0) THEN
          IBL(1)=1
        ELSE
          IF(Y1L .LE. YB(2)) THEN
            IBL(1)=1
          ELSE
            IBL(1)=2
          END IF
        END IF
      ELSE
        CALL CHOPXY(2,1,XMINI)
        IF(ISHAP(2) .EQ. 0) THEN
          IBL(2)=1
        ELSE
          IF(Y2L .LE. YB(2)) THEN
            IBL(2)=1
          ELSE
            IBL(2)=2
          END IF
        END IF
      END IF
C
C FIND OUT IF POINTS ARE OUTSIDE INNER CIRCLE.
C (N.B. LINES ARE NOW LIMITED TO FIRST QUADRANT)
   40 OUTC1=CIRCL1(X1L,Y1L,RI2)
      OUTC2=CIRCL1(X2L,Y2L,RI2)
C CASES BOTH POINTS INSIDE OR BOTH POINTS OUTSIDE INNER CIRCLE
      IF(.NOT. OUTC1 .AND. .NOT. OUTC2) THEN
        GO TO 160
      ELSE
        IF(OUTC1 .AND. OUTC2 .AND. THL .GE. 0.0)GO TO 165
      END IF
C
C LOCATE INTERSECTION WITH INNER CIRCLE
      CALL CIRCL2(RI2,INTC1,INTC2,XC1,YC1,XC2,YC2)
      IF(OUTC1 .AND. OUTC2) THEN
C CASE BOTH POINTS OUTSIDE INNER CIRCLE
        IF(.NOT. INTC1) THEN
          GO TO 165
```

```
CIRCOR


          ELSE
            GO TO 50
          END IF
        END IF
C CASES ONE POINT IN AND THE OTHER OUT INNER CIRCLE
        IF(OUTC1) THEN
          CALL CHOPC(2,XC1,YC1)
          IBL(2)=6
        ELSE
          CALL CHOPC(1,XC2,YC2)
          IBL(1)=6
        END IF
C
        GO TO 165
C
C CASE OF BOTH POINTS OUTSIDE INNER CIRCLE AND EXISTENCE OF INTERSECTION
C WITH THIS CIRCLE (I.E. LINE SEPARATED IN TWO SEGMENTS).
   50 SEPAR=.TRUE.
          XTEMP=X2L
          YTEMP=Y2L
          CUTEMP=CUTL(2)
          IBTEMP=IBL(2)
        X2L=XC1
        Y2L=YC1
        CUTL(2)    =1
        IBL(2)=6
        GO TO 60
C
   55 CONTINUE
C (N.B. THE ENTRIES ORIGINALLY IN LOCATION JJ ARE WRITTEN OVER BY THE
C  ENTRIES RELATED TO THE SECOND SEGMENT OF LINE J)
        SEPAR=.FALSE.
        JJ=JJ+1
        X1L=XC2
        Y1L=YC2
        X2L=XTEMP
        Y2L=YTEMP
        CUTL(2)=CUTEMP
        IBL(2)=IBTEMP
        CUTL(1)    =1
        IBL(1)=6
C
   60 TRL=SQRT((X1L-X2L)**2 + (Y1L-Y2L)**2)
        GO TO 165
C
  160 ELIM=.TRUE.
        X1L=999.
        JO=JO+1
  165 IF(IPRT .EQ. 1)GO TO 170
        WRITE(2,166) J,JFRA,TRL,DEGRE(J),X1L,Y1L,X2L,Y2L
  166 FORMAT(2I6,2X,6F10.3)
  170 X1(JJ,I)=X1L
        IF(ELIM) GO TO 190
C
C REENTER VALUES IN CORRESPONDING ARRAYS.
        SUMLE=SUMLE+TRL
        Y1(JJ,I)=Y1L
```

```
CIRBDR

      X2(JJ,I)=X2L
      Y2(JJ,I)=Y2L
      TRACE(JJ)=TRL
      TH(JJ,I)=THL
C
C DEFINE BOUNDARY NODES.
      DO 180 MEND=1,2
      IF(CUTL(MEND) .EQ. 1) THEN
        CALL BNODE(MEND,IB,XORD,YORD,INC1)
      END IF
  180 CONTINUE
C
      CUT1(JJ,I)=CUTL(1)
      CUT2(JJ,I)=CUTL(2)
      IF(SUMLE .GE. DESLEN)GO TO 200
      IF(SEPAR)GO TO 55
  190 IF(JJ .LT. ESTNFR)GO TO 15
C
  200 RETURN
      END
```

CIRCL2

```
C
C*************************************************************************
C
C  THE SUBROUTINE CIRCL2, USED FOR CIRCULAR BOUNDARIES, CHECKS IF
C  INTERSECTION(S) OF A LINE AND A CIRCLE EXIST WITHIN THE RANGE OF THE
C  FINITE LINE, AND CALCULATES THE COORDINATES OF THIS (THESE)
C  INTERSECTION(S). CIRCL2 FINDS THE SOLUTIONS OF A QUADRATIC EQUATION O
C  THE FORM : A*X**2 + B*X + C = 0.
C
C                         LOCAL VARIABLES
C
C  B & C :   VALUE OF B & C IN THE QUADRATIC EQUATION
C  DISCR :   VALUE OF THE DISCRIMINANT, I.E. B**2-4*A*C, IN THE SOLUTION
C  FOURAC :  VALUE OF THE EXPRESSION 4*A*C.
C            OF THE QUADRATIC EQUATION.
C  OTAN :    VALUE OF 1/TH(J,I) FOR FRACTURE J IN SET I.
C  TWOA :    VALUE OF THE EXPRESSION 2*A
C  YOTAN :   VALUE OF Y1(J,I)/TH(J,I) FOR FRACTURE J IN SET I.
C
      SUBROUTINE CIRCL2(R2,INTC1,INTC2,X1,Y1,X2,Y2)
C
      INTEGER CUTL(2)
      LOGICAL INTC1,INTC2
      COMMON /AREA12/ X1L,Y1L,X2L,Y2L,TRL,THL,CUTL
      COMMON /AREA14/ XCENT,YCENT
C
      INTC1=.FALSE.
      INTC2=.FALSE.
      OTAN=1/THL
      TWOA=2+2*OTAN*OTAN
      YOTAN=Y1L*OTAN
      B=2*OTAN*(X1L-YOTAN-XCENT)-2*YCENT
      B2=B*B
      C=2*YOTAN*(XCENT-X1L)-2*X1L*XCENT+X1L*X1L+XCENT*XCENT
     *+YCENT*YCENT+YOTAN*YOTAN-R2
      FOURAC=2*TWOA*C
      DISCR=B2-FOURAC
      IF(DISCR .LE. 0.0)GO TO 200
      DISCR=SQRT(DISCR)
C
C  COMPUTE THE COORDINATES OF THE FIRST INTERSECTION AND CHECK IF WITHIN
C  THE RANGE OF THE FINITE LINE.
      Y1=(-B+DISCR)/TWOA
      IF(Y1 .GE. Y1L .OR. Y1 .LE. Y2L)GO TO 50
      X1=X1L+OTAN*(Y1-Y1L)
      INTC1=.TRUE.
C
C  DO THE SAME FOR SECOND INTERSECTION
   50 Y2=(-B-DISCR)/TWOA
      IF(Y2 .GE. Y1L .OR. Y2 .LE. Y2L)GO TO 200
      X2=X1L+OTAN*(Y2-Y1L)
      INTC2=.TRUE.
C
  200 RETURN
      END
```

BNODE

```
C*********************************************************************
C
C THE SUBROUTINE BNODE IS CALLED BY THE SUBROUTINES RECBDR OR CIRBDR
C TO SET THE VALUE OF NBN AND OF THE INTERSECTION NUMBER (L), AND TO
C RECORD THE COORDINATES AND INCIDENCE OF INTERSECTION L.
C
      SUBROUTINE BNODE(MEND,IB,XORD,YORD,INC1)
C
      INTEGER CUTL(2)
      COMMON /AREA2/ NSET,MAXFRA,MAXINT,MAXELT,NUMINT,NUMNPT,NUMELT,
     *NOINT,IPRT
      COMMON /AREA7/ R(8),XB(8),YB(8),ISHAP(8),IBC(8)
      COMMON /AREA12/ X1L,Y1L,X2L,Y2L,TRL,THL,CUTL
      COMMON /AREA15/ JSET,JFRA,JO,NBN,IBL(2)
      COMMON /AREA16/ LINT
      DIMENSION XORD(MAXINT),YORD(MAXINT)
      INTEGER INC1(MAXINT,2),IB(MAXINT)
      EQUIVALENCE (LINT,L)
C
      IF(IBC(IBL(MEND)) .GT. 0) CUTL(MEND)=2
C
      L=L+1
      NBN=NBN+1
      IF(MEND .EQ. 1) THEN
        XL=X1L
        YL=Y1L
      ELSE
        XL=X2L
        YL=Y2L
      END IF
      XORD(L)=XL
      YORD(L)=YL
      INC1(L,1)=JSET
      INC1(L,2)=JFRA
      IB(L)=IBL(MEND)
C
      RETURN
      END
```

CHOPC

```
C
C**********************************************************************
C
C  THE SUBROUTINE CHOPC IS CALLED BY THE SUBROUTINE CIRBDR TO COMPUTE
C  THE CORRECTED LENGTH AND THE NEW COORDINATES OF THE CHOPPED OUT
C  EXTREMITY OF A LINE GOING ACROSS A CIRCULAR BOUNDARY.
C
      SUBROUTINE CHOPC(MEND,XI,YI)
C
      INTEGER CUTL(2)
      COMMON /AREA12/ X1L,Y1L,X2L,Y2L,TRL,THL,CUTL
C
      CUTL(MEND)=1
      IF(MEND .EQ. 1) THEN
        X=X1L
        Y=Y1L
      ELSE
        X=X2L
        Y=Y2L
      END IF
      DIFLE=SQRT((X-XI)**2+(Y-YI)**2)
      TRL=TRL-DIFLE
C
      IF(MEND .EQ. 1) THEN
        X1L=XI
        Y1L=YI
      ELSE
        X2L=XI
        Y2L=YI
      END IF
C
      RETURN
      END
```

CHOPXY

```
C*********************************************************************
C
C  THE SUBROUTINE CHOPXY IS CALLED BY THE SUBROUTINES RECBDR OR CIRBDR
C  TO COMPUTE THE CORRECTED LENGTH AND THE NEW COORDINATES OF THE
C  CHOPPED-OUT EXTREMITY OF A LINE GOING ACROSS A MODEL BOUNDARY.
C
C                          LOCAL VARIABLES
C
C  IXY :     KEY DESIGNING WHICH AXIS THE CONSIDERED BOUNDARY IS CROSSING
C            1, FOR A BOUNDARY CROSSING THE X-AXIS,
C            2, FOR A BOUNDARY CROSSING THE Y-AXIS.
C
      SUBROUTINE CHOPXY(MEND,IXY,EXTR)
C
      INTEGER CUTL(2)
      COMMON /AREA12/ X1L,Y1L,X2L,Y2L,TRL,THL,CUTL
C
      CUTL(MEND)=1
      IF(MEND .EQ. 1) THEN
        X=X1L
        Y=Y1L
      ELSE
        X=X2L
        Y=Y2L
      END IF
      IF(IXY .EQ. 1) THEN
        DIFX=X-EXTR
        X=EXTR
        DIFY=DIFX*THL
        Y=Y-DIFY
      ELSE
        DIFY=Y-EXTR
        Y=EXTR
        DIFX=DIFY/THL
        X=X-DIFX
      END IF
      TRL=TRL-SQRT(DIFX*DIFX+DIFY*DIFY)
C
      IF(MEND .EQ. 1) THEN
        X1L=X
        Y1L=Y
      ELSE
        X2L=X
        Y2L=Y
      END IF
      RETURN
      END
```

```
C ***************************************************************************
C
C                       LOCAL VARIABLES (FOR SPCNG1 & SPCNG2)
C
C CORR : CORRECTION FACTOR ACCOUNTING FOR RELATIVE ORIENTATION OF MODEL
C        PLANE (ALSO SAMPLING LINE IN SPCNG2) AND FRACTURE SET BEING
C        SAMPLED.
C IL : COUNTING INDEX FOR SAMPLING LINE.
C INT : INDEX FOR COUNTING THE INTERSECTIONS ON THE CURRENT SAMPLING
C       LINE WITH THE CURRENT FRACTURE SET.
C IR(INT) : PERMUTATION MADE ON YINT(INT) BY SUBROUTINE VSRTR.
C JFRAC(INT) : FRACTURE NUMBER IN THE SET BEING SAMPLED, CORRESPONDING
C              TO THE INTERSECTION INT ON THE CURRENT SAMPLING LINE.
C MAXIL : MAXIMUM NUMBER OF INTERSECTIONS OF A SAMPLING LINE WITH
C         FRACTURES OF THE SAME SET.
C NSP : NUMBER OF SPACING VALUES GENERATED THUS FAR FOR THE FRACTURE
C       SET BEING SAMPLED ( NSP(I) FOR SET I IN SPCNG2).
C NTEN : NUMBER OF TIMES 10 SAMPLING LINES HAVE BEEN GENERATED ( FOR
C        THE CURRENT FRACTURE SET IN SPCNG1 ).
C SPAC(NSP) : SPACING VALUE.
C TANLIN : TANGENT OF THE ANGLE OF THE CURRENT SAMPLING LINE.
C XINT(INT) & YINT(INT) : COORDINATES OF INTERSECTION INT ON THE
C            CURRENT SAMPLING LINE ( WITH THE CURRENT FRACTURE SET IN
C            SPCNG2)
C XP(IL) & YP(IL) : COORDINATES OF THE POINT USED TO GENERATE THE
C                   SAMPLING LINE IL.
C
C
C ***************************************************************************
C
C THE SUBROUTINE SPCNG1 COMPUTE 150 SPACING VALUES FOR EACH FRACTURE
C SET. THE SAMPLING LINES ARE WITHIN THE SIMULATION PLANE. THEY ARE
C PERPENDICULAR TO THE AVERAGE ORIENTATION OF THE LINES OF INTERSECTION
C OF THE FRACTURES OF THE SET BEING SAMPLED.
C
      SUBROUTINE SPCNG1(TANTH,X1,Y1,X2,Y2)
C
      DOUBLE PRECISION SEED,PI,RAD
      INTEGER THDSTR
      COMMON /AREA1B/ XMINI,YMAXI,XMAXI,YMINI,IGEOM
      COMMON /AREA2/ NSET,MAXFRA,MAXINT,MAXELT,NUMINT,NUMNPT,NUMELT,
     *NOINT,IPRT
      COMMON /AREA3A/ SEED,PI,RAD,COEF
      COMMON /AREA3B/ TITLE(20)
      COMMON /AREA4/ ATH(5),BTH(5),ATHP(5),BTHP(5),THDSTR(5),NFRAC(5)
      DIMENSION TANTH(MAXFRA,NSET),X1(MAXFRA,NSET),Y1(MAXFRA,NSET),
     *X2(MAXFRA,NSET),Y2(MAXFRA,NSET)
      DIMENSION XP(10),YP(10),XINT(70),YINT(70),JFRAC(70),IR(70)
      DATA MAXIL/70/
C
      WRITE(3,10) TITLE
10    FORMAT('1',//,1X,20A4,//,' FILE: SPACING1.U03')
      DO 250 I=1,NSET
      A=ATHP(I) * RAD
      CORR=SIN(A)
```

```
      THSL = ATH(I) + 90.0
C
C MAKE SURE THAT SAMPLING LINE IS NOT SUB-PARALLEL TO X-AXIS
      IF((THSL .GE. 0.01 .AND. THSL .LE. 179.99) .OR. (THSL .LE. -0.01)
     *.OR. (THSL .GE. 180.01))GO TO 15
      THSL=0.01
   15 THSL=THSL*RAD
      TANLIN = 1 / TAN(THSL)
      NFRA = NFRAC(I)
      NSP = 1
      NTEN = 1
      WRITE(3,20) I
   20 FORMAT(////,' FRACTURE SET ',I4,///,' NUMB',' NTEN','  LINE',
     *4X,'XP',6X,'YP',3X,'FRACT','  SPACING',//)
C
C GENERATE 10 RANDOM POINTS IN THE MODEL.
C
   30 CALL GGUBS(SEED,10,XP)
      CALL GGUBS(SEED,10,YP)
      DO 190 IL=1,10
      XP(IL) = XP(IL)*XMAXI + (1-XP(IL))*XMINI
      YP(IL) = YP(IL)*YMAXI + (1-YP(IL))*YMINI
      INT = 1
      DO 150 J=1,NFRA
      IF(X1(J,I) .GE. 999.0) GO TO 150
      TANFR = 1 / TANTH(J,I)
C
C COMPUTE INTERSECTION ORDINATE YI, AND CHECK FOR YI INSIDE THE RANGE
C FOR THE FRACTURE (J,I).
C
      YI = ((Y1(J,I)*TANFR) - (YP(IL)*TANLIN) - X1(J,I) + XP(IL) )
     * / (TANFR - TANLIN)
      IF(YI .LT. Y2(J,I) .OR. YI .GT. Y1(J,I) ) GO TO 150
C
C COMPUTE INTERSECTION ABSCISSA AND PUT THE COORDINATES IN XINT(40)
C & YINT(40) ARRAYS.
C
      XI = (YI-Y1(J,I)) * TANFR  +  X1(J,I)
      XINT(INT) = XI
      YINT(INT) = YI
      JFRAC(INT) = J
      IR(INT)=INT
      INT = INT+1
      IF(INT .LE. MAXIL) GO TO 150
      WRITE(2,145) INT,NTEN,IL,I
  145 FORMAT(////,' **** ERROR IN SUBROUTINE SPCNG1****',//,
     *' THERE ARE AT LEAST ',I4,' INTERSECTIONS ON SAMPLING LINE ',
     *I4,' *10 +',I4,' OF SET ',I4,//,' INCREASE THE VALUE OF MAXIL '
     *,'AND THE DIMENSION OF XINT, YINT, JFRAC AND IR')
      STOP
  150 CONTINUE
      INT = INT-1
      IF(INT .LE. 1) GO TO 190
C
C ORDER THE INTERSECTIONS BY INCREASING Y VALUES.
C
      CALL VSRTP(YINT,INT,IR)
```

```
SPCNG1

          DO 160 M1=1,INT
          JTEMP=JFRAC(M1)
          JFRAC(M1)=JFRAC(IR(M1))
  160     JFRAC(IR(M1))=JTEMP
          DO 165 M1=1,INT
          XTEMP=XINT(M1)
          XINT(M1)=XINT(IR(M1))
  165     XINT(IR(M1))=XTEMP
C
C CALCULATE SPACING BETWEEN CONSECUTIVE INTERSECTIONS.
C
      DO 175 ISP=2,INT
      DX = XINT(ISP) - XINT(ISP-1)
      DY = YINT(ISP) - YINT(ISP-1)
      SPAC = SQRT(DX*DX + DY*DY) * CORR
      WRITE(3,170) NSP,NTEN,IL,XP(IL),YP(IL),JFRAC(ISP),SPAC
  170 FORMAT(3I5,2F8.2,I5,F10.2)
      NSP = NSP+1
  175 CONTINUE
      IF(NSP-151) 190,250,250
  190 CONTINUE
      IF(NSP-151) 195,250,250
  195 NTEN = NTEN+1
      GO TO 30
  250 CONTINUE
      RETURN
      END
```

```
C ***********************************************************************
C
C THE SUBROUTINE SPCNG2 COMPUTES 150 SPACING VALUES FOR EACH FRACTURE
C SET. THE SAMPLING LINES ARE RANDOMLY ORIENTED WITHIN THE SIMULATION
C PLANE.
C
      SUBROUTINE SPCNG2(TANTH,X1,Y1,X2,Y2)
C
      DOUBLE PRECISION SEED,PI,RAD
      INTEGER THDSTR
      COMMON /AREA1B/ XMINI,YMAXI,XMAXI,YMINI,IGEOM
      COMMON /AREA2/ NSET,MAXFRA,MAXINT,MAXELT,NUMINT,NUMNPT,NUMELT,
     *NOINT,IPRT
      COMMON /AREA3A/ SEED,PI,RAD,COEF
      COMMON /AREA3B/ TITLE(20)
      COMMON /AREA4/ ATH(5),BTH(5),ATHP(5),BTHP(5),THDSTR(5),NFRAC(5)
      DIMENSION TANTH(MAXFRA,NSET),X1(MAXFRA,NSET),Y1(MAXFRA,NSET),
     *X2(MAXFRA,NSET),Y2(MAXFRA,NSET)
      DIMENSION XP(10),YP(10),THSL(10),NSP(5)
      DIMENSION XINT(70),YINT(70),JFRAC(70),IR(70)
      DATA MAXIL/70/,NSP/5*1/
C
      WRITE(4,10) TITLE
10    FORMAT('1',//,1X,20A4,//,' FILE: SPACING2.U04')
      NTEN = 1
C
      WRITE(4,20)
   20 FORMAT(////,'  NTEN',' LINE',4X,'XP',6X,'YP',' ORIENTATION ',
     *'  SET',3X,'ANGLE',3X,'NUMB ','FRACT','  SPACING',//)
C
C GENERATE 10 RANDOM POINTS AND 10 RANDOM ORIENTATIONS.
C
   30 CALL GGUBS(SEED,10,XP)
      CALL GGUBS(SEED,10,YP)
      CALL GGUBS(SEED,10,THSL)
C
C
      DO 300 IL=1,10
      XPIL = XP(IL)*XMAXI + (1-XP(IL))*XMINI
      YPIL = YP(IL)*YMAXI + (1-YP(IL))*YMINI
      TH=THSL(IL)*180.
C
C MAKE SURE THAT SAMPLING LINE IS NOT SUB-PARALLEL TO X-AXIS
      IF((TH .GE. 0.01 .AND. TH .LE. 179.99) .OR. (TH .LE. -0.01)
     *.OR. (TH .GE. 180.01)) GO TO 35
      TH=0.01
   35 TH=TH*RAD
      TANLIN=1/TAN(TH)
C
C
      DO 250 I=1,NSET
        IF(NSP(I) .GE. 150)GO TO 250
      INT = 1
      NFRA = NFRAC(I)
      ANGLE = ABS(THSL(IL)-ATH(I))
```

```
SPCNG2

      A1=ANGLE*RAD
      A2=ATHP(I) * PAD
      CORR = SIN(A1) * SIN(A2)
      DO 150 J=1,NFRA
      IF(X1(J,I) .GE. 999.0) GO TO 150
      TANFR = 1 / TANTH(J,I)
      XX1=X1(J,I)
      YY1=Y1(J,I)
C
C COMPUTE INTERSECTION ORDINATE YI, AND CHECK FOR YI INSIDE THE RANGE
C FOR THE FRACTURE (J,I).
C
      YI = ((YY1    *TANFR) - (YPIL  *TANLIN) - XX1      + XPIL    )
     * / (TANFR - TANLIN)
      IF(YI .LT. Y2(J,I) .OR. YI .GT. YY1      ) GO TO 150
C
C COMPUTE INTERSECTION ABSCISSA AND PUT THE COORDINATES IN XINT(40)
C & YINT(40) ARRAYS.
C
      XI = (YI-YY1     ) * TANFR  +  XX1
      XINT(INT) = XI
      YINT(INT) = YI
      JFRAC(INT) = J
      IR(INT)=INT
      INT = INT+1
      IF(INT .LE. MAXIL) GO TO 150
      WRITE(2,145) INT,I,NTEN,IL
  145 FORMAT(////,' **** ERROR IN SUBROUTINE SPCNG2****',/,
     *' THERE ARE AT LEAST ',I4,' INTERSECTIONS WITH FRACTURES OF SET '
     *,I4,' ON SAMPLING LINE ',I4,' *10 +',I4,/,' INCREASE THE VALUE OF
     *MAXIL AND THE DIMENSION OF XINT, YINT, JFRAC AND IR')
      STOP
  150 CONTINUE
      INT = INT-1
      IF(INT .LE. 1) GO TO 250
C
C ORDER THE INTERSECTIONS BY INCREASING Y VALUES.
C
      CALL VSRTR(YINT,INT,IR)
         DO 160 M1=1,INT
         JTEMP=JFRAC(M1)
         JFRAC(M1)=JFRAC(IR(M1))
  160    JFRAC(IR(M1))=JTEMP
         DO 165 M1=1,INT
         XTEMP=XINT(M1)
         XINT(M1)=XINT(IR(M1))
  165    XINT(IR(M1))=XTEMP
C
C CALCULATE SPACING BETWEEN CONSECUTIVE INTERSECTIONS.
C
      DO 175 ISP=2,INT
      DX = XINT(ISP) - XINT(ISP-1)
      DY = YINT(ISP) - YINT(ISP-1)
      SPAC = CORR * SQRT(DX*DX + DY*DY)
      WRITE(4,170) NTEN,IL,XPIL,YPIL,THSL(IL),I,ANGLE,NSP(I),
     *JFRAC(ISP),SPAC
  170 FORMAT(2I5,2F8.2,F8.1,3X,I5,3X,F5.1,2X,I5,I5,F10.2)
```

```
  175 NSP(I) = NSP(I)+1
  250 CONTINUE
C
C CHECK IF ALL NSP(I) ARE .GE. 150
      DO 260 I=1,NSET
  260 IF(NSP(I) .LT. 150)GO TO 300
      GO TO 350
  300 CONTINUE
      NTEN = NTEN+1
      GO TO 30
  350 RETURN
      END
```

```
C ***********************************************************************
C
C THE SUBROUTINE INTERS FINDS ALL THE INTERSECTIONS (OR NODES) BETWEEN
C THE LINE SEGMENTS, COMPUTE THE COORDINATES OF THESE NODES AND RECORD
C THEIR INCIDENCE.
C
      SUBROUTINE INTERS(TANTH,X1,Y1,X2,Y2,INC1,INC2,XORD,YORD,IB)
C
      INTEGER THDSTR
      COMMON /AREA1B/ XMINI,YMAXI,XMAXI,YMINI,IGEOM
      COMMON /AREA2/ NSET,MAXFRA,MAXINT,MAXELT,NUMINT,NUMNPT,NUMELT,
     *NOINT,IPRT
      COMMON /AREA4/ ATH(5),BTH(5),ATHP(5),BTHP(5),THDSTR(5)
     *,NFRAC(5)
      COMMON /AREA15/ LINT
      DIMENSION TANTH(MAXFRA,NSET),X1(MAXFRA,NSET),Y1(MAXFRA,NSET),
     *X2(MAXFRA,NSET),Y2(MAXFRA,NSET)
      DIMENSION XORD(MAXINT),YORD(MAXINT)
      INTEGER INC1(MAXINT,2),INC2(MAXINT,2),IB(MAXINT)
      EQUIVALENCE (LINT,L)
C
C N.B. BOUNDARY NODES HAVE ALREADY BEEN DEFINED IN SUBROUTINE GENLIN AND
C      THE OTHER SUBROUTINES CALLED BY GENLIN.
C
      DO 200 I=1,NSET
        NFRAC1 = NFRAC(I)
        IF(THDSTR(I) .EQ. 2) GO TO 8
          IBEG = I+1
          IF(IBEG .GT. NSET)GO TO 200
          GO TO 9
    8   IBEG = I
    9 DO 195 J=1,NFRAC1
          IF(X1(J,I) .GE. 999.0) GO TO 195
          X1L=X1(J,I)
          Y1L=Y1(J,I)
          X2L=X2(J,I)
          Y2L=Y2(J,I)
          TAN1=1/TANTH(J,I)
C
C  CHECK ALL THE LINES IN THE NETWORK FOR ANY POSSIBLE INTERSECTION WITH
C  LINE (J,I).
C
       DO 188 II=IBEG,NSET
         NFRAC2 = NFRAC(II)
         IF(II .GT. I) GO TO 24
         JBEG = J+1
         IF(JBEG .LE. NFRAC2) GO TO 25
         GO TO 188
   24    JBEG = 1
   25    DO 185 JJ=JBEG,NFRAC2
         IF(X1(JJ,II) .GE. 999.0) GO TO 185
         TAN2 = 1 / TANTH(JJ,II)
         IF(ABS(TAN1-TAN2) .LE. 1DE-6)GO TO 185
C
C  COMPUTE INTERSECTION ORDINATE (YI) , AND CHECK FOR YI INSIDE THE
```

```
C  RANGE FOR BOTH FRACTURES.
C
      YI = ( (Y1L*TAN1)-(Y1(JJ,II)*TAN2)-X1L+X1(JJ,II) )
     *  / (TAN1 - TAN2)
      IF(YI .LT. Y2L .OR. YI .GT. Y1L) GO TO 185
      IF(YI .LT. Y2(JJ,II) .OR. YI .GT. Y1(JJ,II)) GO TO 185
C
C  COMPUTE INTERSECTION ABSCISSA (XI).
C
      XI = (YI-Y1L) * TAN1    + X1L
C
C  PUT THE INTERSECTION COORDINATES INTO THE XORD & YORD ARRAYS,
C  AND RECORD THE INCIDENCE OF THIS INTERSECTION.
C
      L=L+1
      XORD(L) = XI
      YORD(L) = YI
      INC1(L,1) = I
      INC1(L,2) = J
      INC2(L,1) = II
      INC2(L,2) = JJ
  185 CONTINUE
  188 CONTINUE
  195 CONTINUE
  200 CONTINUE
      NUMINT = L
C
      IF(NUMINT .LT. MAXINT) GO TO 210
      WRITE(2,206) NUMINT,MAXINT
  206 FORMAT('1**** ERROR CONCERNING MAXINT ****',///,
     *' THERE ARE ',I4,' INTERSECTIONS  AND THERE IS MEMORY SPACE',
     */,'AVAILABLE FOR ',I4,' (MAXINT)')
      STOP
  210 RETURN
      END
```

EFFINT

```
C ***********************************************************************
C
C THE SUBROUTINE EFFINT DETERMINES THE FLOW-EFFECTIVENESS OF EVERY
C INTERSECTION AND RECORD THE APPROPRIATE VALUE (I.E. 0 OR 1)
C IN THE ARRAY NOEFF.
C
C                     LOCAL VARIABLES
C
C II : NUMBER OF ITERATIONS OF THE COMPLETE LOOP OVER ALL THE FRACTURES
C INTER : NUMBER OF THE LAST EFFECTIVE INTERSECTION FOUND ON A FRACTURE
C NOINT : SUM OF ALL THE NON-EFFECTIVE INTERSECTIONS AFTER THE PREVIOUS
C         ITERATION.
C NOINT2 : SUM OF ALL THE NON-EFFECTIVE INTERSECTIONS AT THE END OF
C          CURRENT ITERATION.
C
      SUBROUTINE EFFINT(X1,INC1,INC2,EFF,XORD,YORD,IB)
      INTEGER THDSTR
      COMMON /AREA1B/ XMINI,YMAXI,XMAXI,YMINI,IGEOM
      COMMON /AREA2/ NSET,MAXFRA,MAXINT,MAXELT,NUMINT,NUMNPT,NUMELT,
     *NOINT,IPRT
      COMMON /AREA4/ ATH(5),BTH(5),ATHP(5),BTHP(5),THDSTR(5),
     *NFRAC(5)
      COMMON /AREA7/ R(8),XB(8),YB(8),ISHAP(8),IBC(8)
      DIMENSION X1(MAXFRA,NSET),XORD(MAXINT),YORD(MAXINT)
      INTEGER INC1(MAXINT,2),INC2(MAXINT,2),IB(MAXINT)
      LOGICAL EFF(MAXINT),EFL
C
      II=1
      NOINT=0
C
C INITIALIZE EFF ARRAY.
      DO 10 LL=1,NUMINT
      IF (IB(LL) .EQ. 0) THEN
        EFF(LL)=.TRUE.
      ELSE
        IF(IBC(IB(LL)) .EQ. 0) THEN
          EFF(LL)=.FALSE.
        ELSE
          EFF(LL)=.TRUE.
        END IF
      END IF
   10 CONTINUE
C
C BEGINNING OF 3 NESTED DO-LOOPS (SETS > FRACTURES > INTERSECTIONS ).
   20 DO 100 I=1,NSET
        NFRA = NFRAC(I)
      DO 95 J=1,NFRA
           IF(X1(J,I) .GE. 999.0) GO TO 95
C
C COUNT THE NUMBER OF EFFECTIVE INTERSECTIONS (M) ON FRACTURE (J,I),
C UNTIL M=3.
C
      M = 1
      DO 40 LL=1,NUMINT
      IF(INC1(LL,2) .NE. J .AND. INC2(LL,2) .NE. J) GO TO 40
```

```
 EFFINT

      IF(M .EQ. 3) GO TO 95
      IF(.NOT. EFF(LL))GO TO 40
      IF((INC1(LL,1) .EQ. I .AND. INC1(LL,2) .EQ. J)
     *.OR. (INC2(LL,1) .EQ. I .AND. INC2(LL,2) .EQ. J)) THEN
        INTER=LL
        M=M+1
      END IF
   40 CONTINUE
      IF(M .NE. 2) GO TO 95
C
C ASSIGN THE VALUE .FALSE. IN ARRAY EFF FOR AN INTERSECTION THAT IS
C HYDRAULICALLY INEFFECTIVE, I.E. WHEN THE CONSIDERED INTERSECTION
C IS THE ONLY INTERSECTION WITH EFF=.TRUE. FOR ANY ONE OF THE TWO
C INTERSECTING FRACTURES.
C
      LL = INTER
      EFF(LL)=.FALSE.
   95 CONTINUE
  100 CONTINUE
C
C SUM UP THE NOEFF ARRAY AND CHECK FOR INCREASE WITH RESPECT TO
C PREVIOUS ITERATION.
C
      NOINT2 = 0
      DO 110 LL=1,NUMINT
  110 IF(.NOT. EFF(LL)) NOINT2=NOINT2+1
      IF(NOINT .EQ. NOINT2)GO TO 140
  120 NOINT = NOINT2
      II = II+1
       GO TO 20
  140 WRITE(2,145) II,NOINT
  145 FORMAT(////,' AFTER ITERATION ',I3,'  OF COMPLETE LOOPING OVER ALL
     *THE FRACTURES,'/' THE TOTAL NUMBER OF NON-EFFECTIVE INTERSECTIONS
     *IS FOUND TO BE ',I4)
C
C  REARRANGE THE INTERSECTIONS ARRAYS, PLACING THE EFFECTIVE
C  INTERSECTIONS FIRST.
C
      M=1
      DO 250 LL=1,NUMINT
      IF(.NOT. EFF(LL))GO TO 250
      IF(M .EQ. LL) GO TO 240
      IBL = IB(LL)
      XL = XORD(LL)
      YL = YORD(LL)
      I11L = INC1(LL,1)
      I12L = INC1(LL,2)
      I21L = INC2(LL,1)
      I22L = INC2(LL,2)
      EFL = EFF(LL)
C
      IB(LL) = IB(M)
      XORD(LL) = XORD(M)
      YORD(LL) = YORD(M)
      INC1(LL,1) = INC1(M,1)
      INC1(LL,2) = INC1(M,2)
      INC2(LL,1) = INC2(M,1)
```

```
      INC2(LL,2) = INC2(M,2)
      EFF(LL) = EFF(M)
C
      IB(M) = IBL
      XORD(M) = XL
      YORD(M) = YL
      INC1(M,1) = I11L
      INC1(M,2) = I12L
      INC2(M,1) = I21L
      INC2(M,2) = I22L
      EFF(M) = EFL
C
  240 M = M+1
  250 CONTINUE
      NUMNPT = M-1
C
C PRINT OUT INTERSECTION DATA.
C
      CALL OUTPT1(XORD,YORD,IB,INC1,INC2)
C
      RETURN
      END
```

OUTPT1

```
C
C ***************************************************************************
C
C THE SUBROUTINE OUTPT1 CREATES BOTH A FORMATTED AND AN UNFORMATTED
C FILE CONTAINING THE INTERSECTION DATA.
C
      SUBROUTINE OUTPT1(XORD,YORD,IB,INC1,INC2)
C
      COMMON /AREA2/ NSET,MAXFRA,MAXINT,MAXELT,NUMINT,NUMNPT,NUMELT,
     *NOINT,IPRT
      COMMON /AREA3B/ TITLE(20)
      DIMENSION XORD(NUMNPT),YORD(NUMNPT)
      INTEGER INC1(MAXINT,2),INC2(MAXINT,2),IB(NUMNPT)
C (N.B. INC1 AND INC2 HAVE TWO COLUMNS : FIRST COLUMN CONTAINS
C       FRACTURE SET NUMBER, SECOND ONE THE FRACTURE NUMBER.
C       STORAGE IS BY COLUMN)
C
      WRITE(5) NUMNPT,MAXINT
      WRITE(5) XORD
      WRITE(5) YORD
      WRITE(5) IB
      WRITE(5) INC1
      WRITE(5) INC2
C
      WRITE(2,10) NUMINT,NUMNPT
   10 FORMAT(///,' TOTAL NUMBER OF INTERSECTIONS (NUMINT) :',I5,/,
     *' NUMBER OF EFFECTIVE INTERSECTIONS (NUMNPT) :',I5)
      IF(IPRT .EQ. 1)GO TO 100
C
      WRITE(15,18) TITLE
18    FORMAT('1',//,1X,20A4,//,' FILE: NODFO.U15')
      WRITE(15,20) NUMNPT,NOINT,MAXINT
   20 FORMAT(////,25X,'INTERSECTION DATA',///,' NUMBER OF INTERSECTIONS',
     *//10X,'EFFECTIVE :',T30,I5,' (NUMNPT)',/,10X,'NON-EFFECTIVE :',T30,
     *I5,' (NOINT)',//,10X,'MAXINT=',T30,I5,     //,' FILE CONTAINS :',/
     *10X,'XORD(NUMNPT), FORMAT(10F10.6)',//,10X,'YORD(NUMNPT), FORMAT(10
     *F10.6)',//,10X,'IB(NUMNPT), FORMAT(50I2)',//,10X,'INC1(MAXINT,2), '
     *,'FORMAT(20I5)',//,10X,'INC2(MAXINT,2), FORMAT(20I5)')
      WRITE(15,25) XORD
      WRITE(15,25) YORD
   25 FORMAT(10F10.6)
      WRITE(15,30) IB
   30 FORMAT(50I2)
      WRITE(15,35) INC1
      WRITE(15,35) INC2
   35 FORMAT(20I5)
C
  100 RETURN
      END
```

PLOTFR

```
C  ************************************************************************
C
C THE SUBROUTINE PLOTFR CREATES A PLOT OF THE LINE NETWORK.
C
      SUBROUTINE PLOTFR(PLTSIZ,TANTH,X1,Y1,X2,Y2,XORD,YORD)
C
      INTEGER THDSTR
      COMMON /AREA1A/ XMINO,YMAXO,XMAXO,YMINO
      COMMON /AREA1B/ XMINI,YMAXI,XMAXI,YMINI,IGEOM
      COMMON /AREA2/ NSET,MAXFRA,MAXINT,MAXELT,NUMINT,NUMNPT,NUMELT,
     *NOINT,IPRT
      COMMON /AREA3B/ TITLE(20)
      COMMON /AREA4/ ATH(5),BTH(5),ATHP(5),BTHP(5),THDSTR(5),
     *NFRAC(5)
      COMMON /AREA7/ R(8),XB(8),YB(8),ISHAP(8),IBC(8)
      DIMENSION TANTH(MAXFRA,NSET),X1(MAXFRA,NSET),
     *Y1(MAXFRA,NSET),X2(MAXFRA,NSET),Y2(MAXFRA,NSET)
      DIMENSION XORD(MAXINT),YORD(MAXINT)
      CHARACTER Z*8
C
C  INITIALIZE AND SCALE THE PLOT.
C
      WRITE(Z,5) TITLE(1)
    5 FORMAT(1A4)
      READ(Z,7) RNUM
    7 FORMAT(1X,F3.0)
C
      XLEN = (XMAXO-XMINO)
      YLEN = (YMAXO-YMINO)
      FCTR = (PLTSIZ-3.)/ XLEN
C
      DELTAV = 1.0
      XOR=2.0 - XMINO*FCTR
      YOR=2.0 - YMINO*FCTR
C
      CALL PLOTS(0,0,99)
      CALL PLOT(XOR,YOR,-3)
      CALL FACTOR(FCTR)
C
      XTITLE = (XMAXO+XMINO)*0.5 - 0.15*PLTSIZ/FCTR
      YTITLE = YMAXO + 0.05*PLTSIZ/FCTR
      SIZE = 0.025*PLTSIZ/FCTR
      CALL SYMBOL(XTITLE,YTITLE,SIZE,'REALIZATION ',0.0,12)
      CALL NUMBER(999.0,999.0,SIZE,RNUM,0.0,-1)
C
C DRAW THE EXTERNAL BOUNDARIES.
C
      CALL AXIS(XMINO,YMINO,'X-AXIS',-6,XLEN,0.0,XMINO,DELTAV)
      CALL AXIS(XMAXO,YMINO,6H      ,-1,YLEN,90.0,YMINO,DELTAV)
      CALL AXIS(XMINO,YMINO,'Y-AXIS',6,YLEN, 90.0,YMINO,DELTAV)
      CALL AXIS(XMINO,YMAXO,6H      ,0,XLEN,0.0,XMINO,DELTAV)
C
C  DRAW THE INTERNAL BOUNDARIES.
C
      IF(IGEOM .EQ. 1) THEN
```

```
 PLOTFR

          CALL PLOT(XMINI,YMINI,3)
          CALL PLOT(XMAXI,YMINI,2)
          CALL PLOT(XMAXI,YMAXI,2)
          CALL PLOT(XMINI,YMAXI,2)
          CALL PLOT(XMINI,YMINI,2)
        ELSE
          WRITE(2,25)
   25     FORMAT('1 ****** AVERTISSEMENT DE PLOTFR ******',///,
       *  'LA SOUS-ROUTINE CIRCL N''EXISTE PAS A L''U2AC :',
       *  'LES LIMITES CIRCULAIRES NE SONT PAS TRACEES',///)
          X=XB(1)
          Y=YB(1)
          CALL PLOT(X,Y,3)
          X=XB(3)
          Y=YB(3)
          CALL PLOT(X,Y,2)
C         RADIUS=R(3)
C         CALL CIRCL(X,Y,90.,0.,RADIUS,RADIUS,0.5)
             X=XB(4)
             Y=YB(4)
             CALL PLOT(X,Y,3)
          X=XB(6)
          Y=YB(6)
          CALL PLOT(X,Y,2)
C         RADIUS=R(6)
C         CALL CIRCL(X,Y,0.,90.,RADIUS,RADIUS,0.5)
        END IF
C
C  DRAW ALL THE LINE SEGMENTS (AND, OPTIONNALLY, WRITE DOWN THEIR NUMBER
C
        SIZE = 0.25 / FCTR
        MESS1 = 51
        MESS2 = 50
        DO 100  I=1,NSET
           NFRA = NFRAC(I)
        DO 90 J=1,NFRA
          IF(X1(J,I) .GE. 999.0) GO TO 90
        XX1 = X1(J,I)
        YY1 = Y1(J,I)
        XX2 = X2(J,I)
        YY2 = Y2(J,I)
        CALL PLOT(XX1,YY1,3)
        CALL PLOT(XX2,YY2,2)
        IF(NUMNPT .GT. 200 .OR. IPRT .EQ. 1)GO TO 90
C
          RJ = FLOAT(J)
          X = ( (XX1+XX2)*0.5)
          Y = ( (YY1+YY2)*0.5) - SIZE
          IF(ABS(TANTH(J,I)) .LT. 1) GO TO 30
          CALL SYMBOL(X,Y,SIZE,MESS1,0.0,-1)
          GO TO 40
   30     CALL SYMBOL(X,Y,SIZE,MESS2,0.0,-1)
   40     CALL NUMBER(999.0,999.0,SIZE,RJ,0.0,-1)
C
   90 CONTINUE
  100 CONTINUE
C
```

```
C  CIRCLE (AND, OPTIONALLY, WRITE THE NUMBER) OF THE EFFECTIVE
C  INTERSECTIONS, OR NODES.
C
      MESS = 1
      SIZE = 0.15 / FCTR
      DO 200 K=1,NUMNPT
      RNUM = K
      X =  XORD(K)
      Y =  YORD(K)
      CALL SYMBOL(X,Y,SIZE,MESS,0.0,-1)
      IF(NUMNPT .GT. 200 .OR. IPRT .EQ. 1)GO TO 200
C
        XN = X + SIZE
        YN = Y + SIZE
        CALL NUMBER(XN,YN,SIZE,RNUM,0.0,-1)
C
  200 CONTINUE
C
C
      ENDX = XMAXO+2.
      ENDY=-YOR/FCTR
      CALL PLOT(ENDX,0.00,999)
      RETURN
      END
```

ELTDEF

```
C ***********************************************************************
C
C  THE SUBROUTINE ELTDEF DEFINES THE ELEMENTS IN ALL THE FRACTURES
C  WITH MORE THAN ONE EFFECTIVE INTERSECTIONS.
C
C                    LOCAL  VARIABLES
C
C APER(J) : APERTURE OF FRACTURE J IN THE SET BEING PROCESSED.
C INTER(M) : NUMBER OF THE INTERSECTION (GENERAL NUMBERING)
C              CORRESPONDING TO INTERSECTION M IN THE CURRENTLY
C              PROCESSED FRACTURE.
C MAXEFF : MAXIMUM NUMBER OF EFFECTIVE INTERSECTIONS ON A SINGLE
C          FRACTURE
C ML : COUNTER FOR FLOW-EFFECTIVE INTERSECTIONS ON LINE CURRENTLY
C       PROCESSED.
C NINTER : NUMBER OF EFFECTIVE INTERSECTIONS ON THE CURRENTLY
C          PROCESSSED FRACTURE.
C
      SUBROUTINE ELTDEF(X1,Y1,X2,Y2,INC1,INC2,XORD,YORD,APER,CUT1,CUT2,
     *TH,THP,ELOR,ELORP,ELLEN,W,NOD1,NOD2,ISET)
C
      DOUBLE PRECISION SEED,PI,RAD
      INTEGER APDSTR,THDSTR
      COMMON /AREA1B/ XMINI,YMAXI,XMAXI,YMINI,IGEOM
      COMMON /AREA2/ NSET,MAXFRA,MAXINT,MAXELT,NUMINT,NUMNPT,NUMELT,
     *NOINT,IPRT
      COMMON /AREA3A/ SEED,PI,RAD,COEF
      COMMON /AREA4/ ATH(5),BTH(5),ATHP(5),BTHP(5),THDSTR(5),
     *NFRAC(5)
      COMMON /AREA6/ AAP(5),BAP(5),APDSTR(5),DAAP,NAAP,IAAP,NAP,IAP
C
      DIMENSION EFFLE(5),EFFVO(5),DEADLE(5),DEADVO(5)
      DIMENSION YINT(100),INTER(100)
      INTEGER CUT1(MAXFRA,NSET),CUT2(MAXFRA,NSET)
      DIMENSION TH(MAXFRA,NSET),THP(MAXFRA,NSET)
      DIMENSION X1(MAXFRA,NSET),Y1(MAXFRA,NSET),X2(MAXFRA,NSET),
     *Y2(MAXFRA,NSET)
      DIMENSION APER(MAXFRA)
      DIMENSION XORD(MAXINT),YORD(MAXINT)
      INTEGER INC1(MAXINT,2),INC2(MAXINT,2)
      DIMENSION ELLEN(MAXELT),W(MAXELT),ELOR(MAXELT),ELORP(MAXELT)
      INTEGER NOD1(MAXELT),NOD2(MAXELT),ISET(MAXELT)
      DATA MAXEFF/100/
C
C
      RADI = 1./RAD
      DO 350 IAAP=1,NAAP
      IF(IAAP .EQ. 1)GO TO 12
        DO 10 I=1,NSET
   10   AAP(I)=AAP(I)+DAAP
C
C PRINT OUT INPUT APERTURE DATA
   12 WRITE(2,14) IAAP,(I, I=1,NSET)
   14 FORMAT('1',///,10X,'APERTURE AND POROSITY DATA',
     *//////////,10X,'INPUT DATA FOR APERTURE DISTRIBUTIONS',10X,
```

```
     *'IAAP=',I4,///,' FRACTURE SET :',T25,5I10,///)
      WRITE(2,16) (APDSTR(I), I=1,NSET)
   16 FORMAT(//' DISTRIBUTION TYPE :',T25,5I10)
      WRITE(2,18) (AAP(I), I=1,NSET)
   18 FORMAT(/' DISTRIBUTION PARAMETERS',//,10X,'A :',T25,1P5E10.2)
      WRITE(2,20) (BAP(I), I=1,NSET)
   20 FORMAT(10X,'B :',T25,5F10.2)
C
      DO 300 IAP=1,NAP
        DO 22 I=1,5
        EFFLE(I)=0.0
        EFFVO(I)=0.0
        DEADLE(I)=0.0
        DEADVO(I)=0.0
   22   CONTINUE
        NUMELT=1
C
      DO 200 I=1,NSET
         NFRA = NFRAC(I)
C
C  GENERATE FRACTURE APERTURES.
C
      IF(APDSTR(I) .EQ. 2) GO TO 25
      DO 24 J=1,NFRA
   24 APER(J) = AAP(I)
      GO TO 30
   25 CALL GGNLG(SEED,NFRA,AAP(I),BAP(I),APER)
C
C
   30 DO 195 J=1,NFRA
      IF(X1(J,I) .GE. 999.0) GO TO 195
      APJ = APER(J)
C
C  MAKE A LIST OF ALL THE EFFECTIVE INTERSECTIONS ON FRACTURE (I,J).
C
      ML=1
      DO 60 M=1,NUMNPT
      IF(INC1(M,2) .NE. J .AND. INC2(M,2) .NE. J) GO TO 60
      IF((INC1(M,1) .NE. I .OR. INC1(M,2) .NE. J) .AND.
     *  (INC2(M,1) .NE. I .OR. INC2(M,2) .NE. J))GO TO 60
      INTER(ML) = M
      YINT(ML) = YCRD(M)
      ML = ML+1
      IF(ML .LE. MAXEFF) GO TO 60
      WRITE(2,55) ML,J,I
   55 FORMAT('1**** ERROR IN SUBROUTINE ELTDEF ****',///,
     *' THERE ARE AT LEAST ',I4,' EFFECTIVE INTERSECTIONS ON FRACTURE '
     *,I4,' OF SET ',I4,//,' INCREASE THE VALUE OF MAXEFF AND THE ',
     *'DIMENSION OF YINT,INTER AND NUM')
      STOP
   60 CONTINUE
      NINTER = ML-1
      IF(NINTER .GT. 1) GO TO 65
C
C  ADD THE LENGTH AND VOLUME OF FRACTURES WITH LESS THAN TWO
C  EFFECTIVES INTERSECTIONS TO THE VALUE OF DEADLE(I) AND DEADVO(I).
C
```

```
      DXL=X1(J,I)-X2(J,I)
      DYL=Y1(J,I)-Y2(J,I)
      DEADL=SQRT(DXL*DXL+DYL*DYL)
      DEADLE(I) = DEADLE(I) + DEADL
      DEADVO(I) = DEADVO(I) + DEADL*APJ
      GO TO 195
C
C  ORDER THE INTERSECTIONS BY INCREASING Y VALUE.
C
   65 CALL VSRTR(YINT,NINTER,INTER)
C
C  ADD THE LENGTH AND THE VOLUME OF DEAD END SEGMENT(S) OF FRACTURE
C  (I,J) TO THE VALUE OF DEADLE(I) AND DEADVO(I).
C
      IF(CUT1(J,I) .LT. 2) THEN
        I1=INTER(NINTER)
        DX=X1(J,I)-XORD(I1)
        DY=Y1(J,I)-YORD(I1)
        DLE=SQRT(DX*DX+DY*DY)
        DEADLE(I)=DEADLE(I)+DLE
        DEADVO(I)=DEADVO(I)+DLE*APJ
      END IF
      IF(CUT2(J,I) .LT. 2) THEN
        I2=INTER(1)
        DX=X2(J,I)-XORD(I2)
        DY=Y2(J,I)-YORD(I2)
        DLE=SQRT(DX*DX+DY*DY)
        DEADLE(I)=DEADLE(I)+DLE
        DEADVO(I)=DEADVO(I)+DLE*APJ
      END IF
C
C  DEFINE AS AN  ELEMENT EVERY SEGMENT BETWEEN TWO CONSECUTIVE NODES.
C
  103 DO 150 M=2,NINTER
      M1 = M-1
      DY=YINT(M)-YINT(M1)
      IF(DY .GT. 0.000001)GO TO 110
        WRITE(2,105) I,J,M1,YINT(M1),M,YINT(M)
  105   FORMAT('1',///,' **** JOB ABANDONED DUE TO ELEMENT PRACTICALLY'
     *,'PARALLEL TO X-AXIS',///,' SET',I3,';  FRACTURE',I5,//,
     *2(' INTERSECTION',I3,', Y=',T25,F10.7,/))
        STOP
C
  110 XL=SQRT((XORD(INTER(M))-XORD(INTER(M1)))**2 + DY*DY)
      EFFLE(I)=EFFLE(I)+XL
      EFFVO(I)=EFFVO(I)+XL*APJ
C
      NOD1(NUMELT)=INTER(M-1)
      NOD2(NUMELT)=INTER(M)
      ELLEN(NUMELT)=XL
      W(NUMELT)=APJ
      ISET(NUMELT)=I
      THL = ATAN(TH(J,I)) * RADI
       IF(THL .LE. 0.0) THL=180.+THL
       ELOR(NUMELT)=THL
      ELORP(NUMELT)=THP(J,I)
C
```

```
ELTDEF


      NUMELT=NUMELT+1
  150 CONTINUE
C
  195 CONTINUE
  200 CONTINUE
      NUMELT = NUMELT-1
      IF(NUMELT .LT. MAXELT)GO TO 204
      WRITE(2,202) NUMELT,MAXELT
  202 FORMAT('1*** ERROR CONCERNING MAXELT ***',///,' THERE ARE ',I4,
     *' ELEMENTS AND THERE IS MEMORY SPACE AVAILABLE FOR ',I4,
     *' (MAXELT)')
      STOP
C
C PRINT OUT ELEMENT DATA
  204 CALL OUTPT2(NOD1,NOD2,ISET,ELLEN,W,ELOR,ELORP)
C
C  PRINT OUT POROSITY DATA.
C
C
      WRITE(2,215) IAAP,IAP
  215 FORMAT(////10X,'SUMMARY OF POROSITY DATA',5X,'IAAP=',I4,5X,'IAP='
     *,I4,///,5X,'SET',10X,'LENGTH'
     *,24X,'VOLUME',/,2(11X,'EFFECTIVE',6X,'DEAD'),/)
      WRITE(2,220) (I,EFFLE(I),DEADLE(I),EFFVO(I),DEADVO(I),I=1,NSET)
  220 FORMAT(5X,I3,3P2F11.2,10X,1P2E10.2)
C
  300 CONTINUE
  350 CONTINUE
      RETURN
      END
```

OUTPT2

```
C
C ***************************************************************************
C
C THE SUBROUTINE OUTPT2 CREATES TWO PAIRS OF FILES : ONE PAIR CONTAINING
C THE ELEMENT INCIDENCES AND LENGTH, THE SECOND THE FRACTURE APERTURES.
C EACH PAIR CONSISTS OF ONE FORMATTED AND ONE UNFORMATTED FILE.
C
      SUBROUTINE OUTPT2(NOD1,NOD2,ISET,ELLEN,W,ELOR,ELORP)
C
      COMMON /AREA2/ NSET,MAXFRA,MAXINT,MAXELT,NUMINT,NUMNPT,NUMELT,
     *NOINT,IPRT
      COMMON /AREA39/ TITLE(20)
      COMMON /AREA6/ AAP(5),BAP(5),APDSTR(5),DAAP,NAAP,IAAP,NAP,IAP
      DIMENSION ELLEN(NUMELT),W(NUMELT),ELOR(NUMELT),ELORP(NUMELT)
      INTEGER NOD1(NUMELT),NOD2(NUMELT),ISET(NUMELT)
C
      IF(IAAP .GT. 1 .OR. IAP .GT. 1)GO TO 105
      WRITE(2,15) NUMELT
   15 FORMAT(//,' NUMBER OF ELEMENTS (NUMELT) :',I5)
      WRITE(6) NUMELT
      WRITE(6) NOD1
      WRITE(6) NOD2
      WRITE(6) ELLEN
      WRITE(6) ISET
      WRITE(6) ELOR
      WRITE(6) ELORP
      WRITE(7) NUMELT,NAAP,NAP
      IF(IPRT .EQ. 1)GO TO 105
C
      WRITE(16,50) TITLE,NUMELT
50    FORMAT('1',//,1X,20A4,//,' FILE: ELEF0.U16',
     +///,25X,'ELEMENT DATA',///,'NUMBER OF ELEMENTS :'
     *,T30,I5,' (NUMELT)',///,' FILE CONTAINS :',//,10X,'NOD1(NUMELT), ',
     *'FORMAT(20I5)',//,10X,'NOD2(NUMELT), FORMAT(20I5)',//,
     *10X,'ISET(NUMELT), FORMAT(50I2)',//,
     * 10X,'ELLEN(NUMELT), FORMAT(1P10E10.3)',//,
     *10X,'ELOR(NUMELT), FORMAT(1P10E10.3)',//,
     *10X,'ELORP(NUMELT), FORMAT(1P10E10.3)')
      WRITE(16,60) NOD1
      WRITE(16,60) NOD2
   60 FORMAT(20I5)
      WRITE(16,62) ISET
   62 FORMAT(50I2)
      WRITE(16,70) ELLEN
   70 FORMAT(1P10E10.3)
      WRITE(16,75) ELOR
      WRITE(16,75) ELORP
   75 FORMAT(10F10.3)
C
      WRITE(17,100) TITLE,NUMELT,NAAP,NAP
100   FORMAT('1',//,1X,20A4,//,' FILE: APEF01.U17',
     +///,25X,'FRACTURE APERTURES',///,' NUMBER OF ',
     *'ELEMENTS :',T30,I5,' (NUMELT)',//,' NUMBER OF DIFF. VAL. FOR ',
     *'FIRST PARAM (AAP) :',T50,I5,' (NAAP)',//,' NUMBER OF ',
     *'REALIZATIONS FOR EACH VAL. OF AAP :',T50,I5,' (NAP)',///,
```

```
      OUTPT2


         *' FILE CONTAINS : NAAP*NAP ARRAYS W(NUMELT), FORMAT(1P10E10.3)')
  105 WRITE(7) W
         IF(IPRT .EQ. 1)GO TO 150
         WRITE(17,130) W
  130 FORMAT(1P10E10.3)
C
  150 RETURN
         END
```

G.2 Listing of the Program APEGEN

APEGEN

```
      PROGRAM APEGEN
C*****************************************************************************
C
C  PROGRAM APEGEN GENERATES FRACTURE APERTURE VALUES FOR THE FRACTURE
C  NETWORK GENERATED BY THE PROGRAM NETWRK
C
C  WRITTEN BY A. ROULEAU, DECEMBER 1984; UPDATE BY A. ROULEAU
C  AND R. MACLEOD, JUNE 1985.
C
C*****************************************************************************
      INTEGER APDSTR
      DOUBLE PRECISION SEED
      COMMON /AREA1/ TITLE(20)
      COMMON /AREA2/ NUMNPT,NUMELT,IPRT
      COMMON /AREA3/ AAP(5),BAP(5),SEED,APDSTR,NSET
      COMMON /AREA4/ DAAP,NAAP,MAP,IAAP,IAP
C
C DIMENSION MAXELT=200
      INTEGER NOD1(200),NOD2(200),ISET(200)
      DIMENSION W(200),WW(200),ELLEN(200),ELOR(200),ELORP(200)
      DIMENSION P(200),Q(200),R(200)
C
C DIMENSION MAXNOD=150
      DIMENSION XORD(150),YORD(150)
C
C CONSTANTS
      MAXELT=200
      MAXNOD=150
C
      READ(21,3) TITLE
3     FORMAT(20A4)
C
C READ GENERAL INPUT DATA
      READ(21,5) IPRT,APDSTR,MAP,NAAP,DAAP,SEED,NSET,IOR
5     FORMAT(4I5,F10.0,F5.0,2I5)
      READ(21,8) (AAP(I),BAP(I),I=1,NSET)
8     FORMAT(2F10.0)
C
C READ NETWORK DATA
      READ(5) NUMNPT
      READ(6) NUMELT
      CALL INPT3(XORD,YORD,NOD1,NOD2,ISET,ELLEN,ELOR,ELORP,W)
      IF(IOR .EQ. 0) GO TO 600
C
C WITHIN THE FOLLOWING DO-LOOP (DO 500) , FOR EACH ELEMENT, THE
C COORDINATES OF THE CENTERPOINT IS DETERMINED AND THE ORIENTATION
C IS COMPUTED IN TERMS OF THE DIRECTION COSINES P,Q,R.
C THESE RESULTS ARE NOT USED IN THE PRESENT VERSION, BUT THEY
C WOULD BE USEFUL FOR INSTANCE TO DETERMINE APERTURE AS A FUNCTION
C OF FRACTURE ORIENTATION AND LOCATION.
C
C LOOP OVER ALL THE ELEMENTS
      DO 500 N=1,NUMELT
C
C COMPUTE DIRECTION COSINES
```

```
      THN=ELOR(N)
      THPN=ELORP(N)
      ASIN=SIN(THPN)
      P(N)=COS(THPN)
      Q(N)=ASIN*SIN(THN)
      R(N)=-ASIN*COS(THN)
C CHECK ON DIRECTION COSINES
      PN=P(N)
      QN=Q(N)
      RN=R(N)
      P2=PN*PN
      Q2=QN*QN
      R2=RN*RN
      TOT=P2+Q2+R2
      WRITE(6,*) N,TOT
C
C DETERMINE THE COORDINATES OF THE ELEMENTS
C  FIRST END POINTS
      ND1=NOD1(N)
      ND2=NOD2(N)
      X1E=XORD(ND1)
      Y1E=YORD(ND1)
      X2E=XORD(ND2)
      Y2E=YORD(ND2)
C  THEN CENTERPOINT
      XCE = (X1E+X2E)/2.0
      YCE = (Y1E+Y2E)/2.0
C NOW IN CYLINDRICAL COORDINATES
      TANR=YCE/XCE
      THR=ATAN(TANR)
      RF=YCE/SIN(THR)
C
C MODIFY EXISTING APERTURE VALUE W(N) OR CREATE A NEW ONE
C
C  (NO OPERATION IS DONE HERE IN THE PRESENT VERSION OF APEGEN)
C
500   CONTINUE
      STOP
C
C CREATE AN ENTIRE ARRAY OF NEW APERTURES.
C
600   CALL APARR(W,WW,ISET)
C
      STOP
      END
```

INPT3

```
C*************************************************************************
C
C  SUBROUTINE INPT3 IS CALLED BY APEGEN-MAIN TO READ IN THE NODE
C  COORDINATE DATA (UNIT 9), THE ELEMENT INCIDENCE DATA (UNIT 10)
C  AND THE INITIAL FRACTURE APERTURE DATA (UNIT 11).
C
C*************************************************************************
C
      SUBROUTINE INPT3(XORD,YORD,NOD1,NOD2,ISET,ELLEN,ELOR,ELORP,W)
      COMMON /AREA2/ NUMNPT,NUMELT,IPRT
      DIMENSION XORD(NUMNPT),YORD(NUMNPT)
      DIMENSION ELLEN(NUMELT),W(NUMELT)
      INTEGER NOD1(NUMELT),NOD2(NUMELT),ISET(NUMELT)
C
      READ(5) XORD
      READ(5) YORD
C
      READ(6) NOD1
      READ(6) NOD2
      READ(6) ELLEN
      READ(6) ISET
      READ(6) ELOR
      READ(6) ELORP
C
      READ(7) DUMMY
      READ(7) W
C
      RETURN
      END
```

```
C*************************************************************************
C
C  SUBROUTINE APARR CREATES AN ENTIRE ARRAY OF NEW APERTURE VALUES.
C
C*************************************************************************
C
      SUBROUTINE APARR(W,WW,ISET)
      COMMON /AREA1/ TITLE(20)
      COMMON /AREA2/ NUMNPT,NUMELT,IPRT
      COMMON /AREA3/ AAP(5),BAP(5),SEED,APDSTR,NSET
      COMMON /AREA4/ DAAP,NAAP,NAP,IAAP,IAP
      INTEGER APDSTR
      INTEGER ISET(NUMELT)
      DOUBLE PRECISION SEED
      DIMENSION W(NUMELT),WW(NUMELT)
C
      WRITE(23,5) TITLE,NUMELT,APDSTR
5     FORMAT(1X,20A4,//,' FILE : APEF02.023',/////,
     +25X,'FRACTURE APERTURES',///,5X,'NUMBER OF ELEMENTS : ',T30,I5,
     +' (NUMELT)',//,5X,'DISTRIBUTION TYPE : ',T30,I5)
      DO 350 IAAP=1,NAAP
      IF(IAAP .EQ. 1) GO TO 12
      DO 10 I=1,NSET
10    AAP(I)=AAP(I)+DAAP
C
12    DO 300 IAP=1,NAP
C
C PRINT OUT INPUT APERTURE DATA
      WRITE(23,14) IAAP,IAP,(I,I=1,NSET)
14    FORMAT(////,10X,'INPUT DATA FOR APERTURE DISTRIBUTIONS',10X,
     +'IAAP=',I4,'  IAP=',I4,///,' FRACTURE SET :',T25,5I10,///)
      WRITE(23,18) (AAP(I),I=1,NSET)
18    FORMAT(/,' DISTRIBUTION PARAMETERS',//,10X,'A :',T25,1P5E10.2)
      WRITE(23,20) (BAP(I), I=1,NSET)
20    FORMAT(10X,'B :',T25,5F10.2)
C
C GENERATE APERTURE VALUES
C
      IF (APDSTR .EQ. 2) GO TO 25
      DO 24 N=1,NUMELT
24    WW(N) = AAP(ISET(N))
      GO TO 30
25    DO 28 I=1,NSET
      CALL GGNLG(SEED,NUMELT,AAP(I),BAP(I),W)
      DO 28 N=1,NUMELT
28    IF(ISET(N) .EQ. I) WW(N)=W(N)
C
30    CALL OUTPT3(WW,ISET)
C
300   CONTINUE
C
350   CONTINUE
C
      RETURN
      END
```

OUTPT3

```
C*******************************************************************************
C
C  SUBROUTINE OUTPT3 WRITES OUT A NEW FILE OF APERTURE VALUES
C  IN UNIT 22 (AND 23).
C
C*******************************************************************************
C
      SUBROUTINE OUTPT3(WW,ISET)
      COMMON /AREA1/ TITLE(20)
      COMMON /AREA2/ NUMNPT,NUMELT,IPRT
      COMMON /AREA4/ DAAP,NAAP,NAP,IAAP,IAP
      DIMENSION WW(NUMELT)
      INTEGER ISET(NUMELT)
C
      IF (IAAP .GT. 1 .OR. IAP .GT. 1) GO TO 105
C
      WRITE(22) NUMELT,NAAP,NAP
105   WRITE(22) WW
C
      IF (IPRT .EQ. 1) GO TO 150
C
      WRITE(23,122)
122   FORMAT(///,20X,'ARRAY ISET, FORMAT(50I2)',///)
      WRITE(23,124) ISET
124   FORMAT(50I2)
      WRITE(23,128)
128   FORMAT(///,20X,'ARRAY WW, FORMAT(1P10E10.3)',///)
      WRITE(23,130) WW
130   FORMAT(1P10E10.3)
C
150   RETURN
      END
```

G.3 Listing of the Program NETFLO

NETFLO

```
      PROGRAM NETFLO
C*****************************************************************************
C
C  THE PROGRAM NETFLO IS A NUMERICAL CODE THAT SOLVES THE STEADY-
C  STATE EQUATIONS FOR FLUID FLOW IN THE NETWORK OF FRACTURES GENERATED
C  BY THE PROGRAM NETWRK FORTRAN.
C
C  WRITTEN BY A. ROULEAU, 1983; UPDATE BY A. ROULEAU AND D. LENTZ,
C  APRIL 1985.
C
C*****************************************************************************
C
      COMMON /AREA2/ MAXNOD,MAXELT,NUMNPT,NUMELT,NC,NF,NA,MAXA,
     *MAXCON,MAXFRE,IPRT,JAP
      COMMON /AREA4/ X3(8),Y3(8),HDB(8),HDE(8),ISHAP(8),IBC(8),IGEOM,NBO
      COMMON /AREA5/ MAXEL2,MAXN04,IDIFF
      COMMON /AREA6/ TITLE(20)
      COMMON /AREA8/ VISC,SPGR
C
C  DIMENSION = MAXELT = 200
      INTEGER NOD1(200),NOD2(200),IDUMMY(200)
      DIMENSION W(200),E(200),ELLEN(200),ELOR(200)
C
C  DIMENSION = MAXEL2 = 2 * MAXELT = 400
      INTEGER JT(400)
C
C  DIMENSION = MAXNOD = 150
      INTEGER IB(150),LC(150),JMEM(150),NODNUM(150),NEWNOD(150),
     *JOINT(150),NEWJT(150)
      DIMENSION XORD(150),YORD(150),PHI(150)
      LOGICAL REOR(150)
C
C  DIMENSION = MAXN04 = 4 * MAXNOD = 600
      INTEGER MEMJT(600)
C
C  DIMENSION = MAXFRE = 110
      INTEGER KDIAG(110)
C
C  DIMENSION = MAXCON = MAXNOD - MAXFRE = 150-110=40
      DIMENSION PHIC(40)
C
C  DIMENSION = MAXA = 2000
      DIMENSION A(2000)
C
C  MAXELT + 1 = 200 + 1 = 201
      EQUIVALENCE (JT(1),NOD1),(JT(201),NOD2)
      EQUIVALENCE (ELOR,IDUMMY)
C
C  CONSTANTS
C
      MAXELT = 200
      MAXNOD = 150
      MAXA = 2000
      MAXFRE = 110
      MAXCON = MAXNOD-MAXFRE
```

```
NETFLO

      MAXEL2=2*MAXELT
      MAXNO4=4*MAXNOD
      VISC=1.002/1000.
      SPGR=9800.
C
      READ(1,3) TITLE
    3 FORMAT(20A4)
C READ IN THE BOUNDARY CONDITIONS.
C
      READ(1,5)IGEOM,NAP,NAAP,IPRT
    5 FORMAT(25X,3I5,10X,I5,/)
        IF(IGEOM .EQ. 1)NBO=8
        IF(IGEOM .EQ. 2)NBO=6
      READ(1,7) (ISHAP(I),XB(I),YB(I),IBC(I),HDB(I),HDE(I),I=1,NBO)
    7 FORMAT(I5,5X,2F5.0,I5,2F5.0)
C
C READ IN THE NETWORK DATA
C
      READ(5) NUMNPT
      READ(6) DUMMY
      READ(7) NUMELT,NAAP,NAP
      CALL INPT1(XORD,YORD,IB,NOD1,NOD2,ELLEN,W,ELOR,IDUMMY)
C
C DEFINE THE BOUNDARY CONDITION FOR ALL THE NODES LOCATED ON A BOUNDARY
C
      CALL BOUND(IB,PHI,XORD,YORD)
C
C OPTIMIZE THE NODE NUMBERING AND DEFINE THE CONDENSATION CODE FOR
C SPARSE MATRIX STORAGE.
C
      CALL OPSTOR(IB,LC,KDIAG,JT,JMEM,MEMJT,NODNUM,NEWNOD,JOINT,NEWJT)
C
C SET UP THE MATRICES USING THE NODAL FLOW EQUATIONS, AND SOLVE FOR THE
C UNKNOWN HEAD VALUES (PHI) USING THE CHOLESKI ALGORITHM.
C
      NAPER=NAP*NAAP
      DO 200 JAP=1,NAPER
      IF(JAP .EQ. 1)GO TO 30
      CALL INPT2(W)
   30 CALL SOLPHI(IB,LC,KDIAG,NOD1,NOD2,XORD,YORD,PHI,PHIC,W,A,
     *E,NODNUM,NEWNOD,REOR,ELLEN)
C
C CALCULATE FLOW IN EACH ELEMENT AND TOTAL FLOW THROUGH THE MODEL.
C
      CALL FLOCAL(NOD1,NOD2,W,ELLEN,E,PHI,IB,ELOR)
C
  200 CONTINUE
      STOP
      END
```

INPT1

```
C
C ***************************************************************************
C
C THE SUBROUTINE INPT1 READS IN THE LINE NETWRK AND FRACTURE APERTURE
C DATA FROM UNFORMATTED FILES CREATED BY NETWRK.
C
      SUBROUTINE INPT1(XORD,YORD,IB,NOD1,NOD2,ELLEN,W,ELOR,IDUMMY)
C
      COMMON /AREA2/ MAXNOD,MAXELT,NUMNPT,NUMELT,NC,NF,NA,MAXA,
     *MAXCON,MAXFRE,IPRT,JAP
      DIMENSION XORD(NUMNPT),YORD(NUMNPT)
      INTEGER IB(NUMNPT)
      DIMENSION ELLEN(NUMELT),W(NUMELT),ELOR(NUMELT)
      INTEGER NOD1(NUMELT),NOD2(NUMELT),IDUMMY(NUMELT)
C
      READ(5) XORD
      READ(5) YORD
      READ(5) IB
C
      READ(6) NOD1
      READ(6) NOD2
      READ(6) ELLEN
      READ(5) IDUMMY
      READ(6) ELOR
C
      ENTRY INPT2(W)
      READ(7) W
C
      RETURN
      END
```

BOUND

```
C**********************************************************************
C
C THE SUBROUTINE BOUND DEFINES THE BOUNDARY CONDITIONS FOR ALL THE NODES
C LOCATED ON THE BOUNDARIES OF THE FLOW MODEL.
C
C                        LOCAL VARIABLES
C
C BEG(L) : ORDINATE (ABSCISSA) OF BEGINNING POINT OF BOUNDARY L, IF
C THIS BOUNDARY IS PARALLEL TO Y-AXIS (TO X-AXIS).
C END(L) : SAME AS BEG(L) FOR END POINT.
C GRAD(L) : IF IBC(L)=2, HEAD GRADIENT ALONG BOUNDARY L
C           IF IBC(L)=3, HEAD DIFF. / DIFF. IN LOG OF THE EXTREM. POINT
C
      SUBROUTINE BOUND(IB,PHI,XORD,YORD)
      COMMON /AREA2/ MAXNOD,MAXELT,NUMNPT,NUMELT,NC,NF,NA,MAXA,
     *MAXCON,MAXFRE,IPRT,JAP
      COMMON /AREA4/ XB(8),YB(8),HDB(8),HDE(8),ISHAP(8),IBC(8),IGEOM,NBO
      COMMON /AREA6/ TITLE(20)
      INTEGER IB(MAXNOD)
      DIMENSION XORD(MAXNOD),YORD(MAXNOD),PHI(MAXNOD)
      DIMENSION BEG(8),END(8),GRAD(8)
C
C DETERMINE END POINT (END( ) ) FOR EACH BOUNDARY
C
      IF(ISHAP(2) .EQ. 0) THEN
        END(1)=YB(3)
      ELSE
        END(1)=YB(2)
        END(2)=YB(3)
      END IF
      IF(IGEOM .EQ. 1)GO TO 10
C
C IF CIRCULAR BOUNDARIES
      IF(ISHAP(5) .EQ. 0) THEN
        END(4)=XB(6)
      ELSE
        END(4)=XB(5)
        END(5)=XB(6)
      END IF
      GO TO 20
C
   10 IF(ISHAP(4) .EQ. 0) THEN
        END(3)=XB(5)
      ELSE
        END(3)=XB(4)
        END(4)=XB(5)
      END IF
      IF(ISHAP(6) .EQ. 0) THEN
        END(5)=YB(7)
      ELSE
        END(5)=YB(6)
        END(6)=YB(7)
      END IF
      IF(ISHAP(8) .EQ. 0) THEN
        END(7)=XB(1)
```

```
      ELSE
        END(7)=XB(8)
        END(8)=XB(1)
      END IF
C
C CALCULATE VALUE OF CONSTANTS BEG( ) AND GRAD( ) FOR EACH BOUNDARY
C FOR WHICH IBC=2 OR 3.
C
   20 DO 50 L=1,NBC
      M1=IBC(L)
      IF(M1 .LT. 2) GO TO 50
      DHD=HDE(L)-HDB(L)
      IF(IGEOM .EQ. 2)GO TO 25
C RECTANGULAR BOUNDARIES
      IF(L .EQ. 1 .OR. L .EQ. 2 .OR. L .EQ. 5 .OR. L .EQ. 6) THEN
        BEG(L)=YB(L)
      ELSE
        BEG(L)=XB(L)
      END IF
      GO TO 30
C CIRCULAR BOUNDARIES
   25 IF(L .EQ. 1 .OR. L .EQ. 2) THEN
        BEG(L)=YB(L)
      ELSE
        BEG(L)=XB(L)
      END IF
C
   30 DLE=END(L)-BEG(L)
      IF(M1 .EQ. 2) THEN
        GRAD(L)=DHD/DLE
      ELSE
        GRAD(L)=DHD/ALOG(END(L)/BEG(L))
      END IF
   50 CONTINUE
C
C DETERMINE HEAD VALUE FOR EACH NODE LOCATED ON A BOUNDARY FOR WHICH
C IBC( )=1, 2 OR 3.
C
      NC=0
      DO 100 M=1,NUMNPT
      IBM=IB(M)
      IF(IBM .EQ. 0)GO TO 100
      M1=IBC(IBM)
      IF(M1 .EQ. 0)GO TO 100
      IF(M1 .EQ. 1) THEN
        PHI(M)=HDB(IBM)
      ELSE
C ASSIGN VALUE FOR XORY
        IF(IGEOM .EQ. 1) THEN
          IF(IBM .EQ. 1 .OR. IBM .EQ.2.OR.IBM.EQ.5.OR.IBM.EQ.6) THEN
            XORY=YORD(M)
          ELSE
            XORY=XORD(M)
          END IF
        ELSE
          IF(IBM .EQ. 1 .OR. IBM .EQ. 2) THEN
            XORY=YORD(M)
```

```
BOUND

          ELSE
            XORY=XCRD(M)
          END IF
        END IF
C
        IF(M1 .EQ. 2) THEN
          PHI(M)=HDB(IBM) + GRAD(IBM)*(XORY-BEG(IBM))
        ELSE
          PHI(M)=HDB(IBM) + GRAD(IBM)*ALOG(XORY/BEG(IBM))
        END IF
      END IF
      NC=NC+1
  100 CONTINUE
      NF=NUMNPT-NC
C
C PRINT OUT BOUNDARY DATA
C
      WRITE(31,200) TITLE
200   FORMAT('1',/,1X,20A4,/,' FILE: SUMFLO.U31',
     +///,10X,'OUTPUT FROM SUBROUTINE BOUND',///)
      WRITE(31,340)NUMNPT,NF,NC
  340 FORMAT(//,10X,'TOTAL NUMBER OF NODES',T40,I5,' (NUMNPT)',/,
     *10X,'NUMBER OF FREE NODES',T40,I5,' (NF)',/,
     *10X,'NUMBER OF CONSTRAINED NODES',T40,I5,' (NC)',////)
C
C PRINT OUT CONSTANTS FOR EACH BOUNDARY
      WRITE(31,350)
  350 FORMAT('  BDRY NO   IBC',7X,'BEG',10X,'END',8X,'***GRADIENT***',//)
      WRITE(31,355) (L,IBC(L),BEG(L),END(L),GRAD(L), L=1,NBO)
  355 FORMAT(2I7,3F13.3)
      IF(NF .LE. MAXFRE .AND. NC .LE. MAXCON)GO TO 370
C
      WRITE(31,365)NF,MAXFRE,NC,MAXCON
  365 FORMAT('1**** ERROR CONCERNING MAXFRE OR MAXCON ****',//,' THERE A
     *RE ',I5,' FREE NODES, AND MAXFRE IS',I5,//,' THERE ARE ',I4,' CONS
     *TRAINED NODES, AND MAXCON IS ',I4)
      STOP
  370 RETURN
      END
```

OPSTOR

```
C*********************************************************************
C
C THE SUBROUTINE OPSTOR OPTIMIZES THE NODE NUMBERING AND DEFINES THE
C CONDENSATION CODES FOR MATRIX STORAGE.
C OPSTOR CALLS THE SUBROUTINES SETUP AND OPTNUM.
C
      SUBROUTINE OPSTOR(IB,LC,KDIAG,JT,JMEM,MEMJT,NODNUM,NEWNOD,JOINT,
     *NEWJT)
      COMMON /AREA2/ MAXNOD,MAXELT,NUMNPT,NUMELT,NC,NF,NA,MAXA,
     *MAXCON,MAXFRE,IPRT,JAP
      COMMON /AREA4/ XB(8),YB(8),MDB(8),MDE(8),ISHAP(8),IBC(8),IGEOM,NBO
      COMMON /AREA5/ MAXEL2,MAXNO4,IDIFF
      COMMON /AREA6/ TITLE(20)
      INTEGER KDIAG(MAXFRE)
      INTEGER IB(MAXNOD),LC(MAXNOD),JT(MAXEL2),
     *JMEM(MAXNOD),MEMJT(MAXNO4),NODNUM(MAXNOD),NEWNOD(MAXNOD),
     *JOINT(MAXNOD),NEWJT(MAXNOD)
C
C DEFINE THE TOPOLOGY OF THE ORIGINAL NODE NUMBERING.
C
      CALL SETUP(JT,JMEM,MEMJT)
C
C PRINT OUT THE DATA CONCERNING THE INITIAL NODE NUMBERING.
      WRITE(31,20) IDIFF
   20 FORMAT(///,10X,'OUTPUT FROM SETUP',///,' INITIAL IDIFF=',I5,//)
      IF(IPRT .LT. 3)GO TO 39
C
      WRITE(34,22) TITLE,IDIFF
22    FORMAT('1',//,1X,20A4,//,' FILE: FLOMAT.U34',
     +//,9X,'OUTPUT FROM SETUP',//,' INITIAL IDIFF=',I5,//)
      WRITE(34,24)
   24 FORMAT(10X,'FORMAT(6(I     JT(I)))',///)
      WRITE(34,26)(I,JT(I), I=1,MAXEL2)
   26 FORMAT(6(5X,2I5))
      WRITE(34,28) (I,JMEM(I),I=1,NUMNPT)
   28 FORMAT(////,10X,'FORMAT(6(I     JMEM(I)))',///,6(2I5,5X))
      WRITE(34,33)
   33 FORMAT(///,10X,'FORMAT(I     4 NODES RELATED TO NODE I, I.E. MEMJT(
     *4*(I-1) +1 TO *4))',//)
      DO 37 I=1,NUMNPT
        I4=4*(I-1)
        WRITE(34,36)I,MEMJT(I4+1),MEMJT(I4+2),MEMJT(I4+3),MEMJT(I4+4)
   36   FORMAT(  I5,2X,4I5,5X)
   37 CONTINUE
C
C RENUMBER THE NODES IN ORDER TO MINIMIZE THE BANDWIDTH OF THE MATRIX A
C
   39 CALL OPTNUM(JT,JMEM,MEMJT,NODNUM,NEWNOD,JOINT,NEWJT)
C
C DEFINE THE CONDENSATION CODE (LC) FOR MATRIX PARTITIONING, AND
C DETERMINE THE VALUES IN THE KDIAG ARRAY FOR STORAGE OF SUBMATRIX A.
C
      LC(1)=0
      MB=IB(NEWNOD(1))
      KDIAG(1)=1
```

```
      IF(M6 .EQ. 0)GO TO 50
      IF(IBC(M6) .GT. 0)LC(1)=1
   50 DO 100 M=2,NUMNPT
      M5=NEWNOD(M)
      M7=IB(M5)
C
C CALCULATE LC(M)
      K=0
      IF(M7 .EQ. 0)GO TO 53
      IF(IBC(M7).GT. 0)K=1
   53 LC(M)=LC(M-1)+K
      LCM=LC(M)
      IF(M7 .EQ. 0)GO TO 54
      IF(IBC(M7).GT. 0)GO TO 100
C
C CALCULATE KDIAG FOR NODE M, IF M IS A FREE NODE.
C FIRST, FIND WHICH ONE OF THE NODES RELATED TO NODE M HAS THE MINIMUM
C INDEX.
   54 NREL=JMEM(M5)
      MINREL=NODNUM(MEMJT(M5*4 - 3) )
      IF(NREL .EQ. 1)GO TO 60
      DO 55 M1=2,NREL
        M2=M5*4-4+M1
        M8=NODNUM(MEMJT(M2))
        IF(MINREL .GT. M8)MINREL=M8
   55   CONTINUE
   60 M3=M-LCM
C M3 INDICATES THE LOCATION OF NODE M ON THE DIAGONAL OF THE SUBMATRIX
C OF FREE NODES (A).
      IF(M3 .LT. 2)GO TO 100
      IF(MINREL .LT. M)GO TO 80
      KDIAG(M3)=KDIAG(M3-1)+1
      GO TO 100
   80 KDIAG(M3)=KDIAG(M3-1)+(M-MINREL)-(LCM-LC(MINREL))+1
  100 CONTINUE
      IF(LC(NUMNPT) .EQ. NC)GO TO 105
      WRITE(31,103) LC(NUMNPT),NC
  103 FORMAT('1**** ERROR CONCERNING CONSTRAINED NODES ****',///,
     *' LC(NUMNPT)=',I4,' NC=',I4)
      STOP
  105 CONTINUE
C
      NA=KDIAG(NF)
C
C PRINT OUT LC AND KDIAG ARRAYS
      WRITE(31,106)NA
106   FORMAT(////,10X,'RESULTS FROM OPSTOR',///,' SIZE OF MATRIX A (NA):',
     *I7)
      IF(IPRT .LT. 3)GO TO 120
C
      WRITE(34,108)
  108 FORMAT(///,10X,'CONDENSATION CODE (LC)', FORMAT(6(I    LC(I)))',//)
      WRITE(34,109) (I,LC(I),I=1,NUMNPT)
  109 FORMAT(6(2I5,5X))
      WRITE(34,112)
  112 FORMAT(///,10X,'VARIABLE BANDWIDTH STORAGE CODE (KDIAG)', FORMAT(6(I
     *  KDIAG(I)))',///)
```

```
OPSTOR

      WRITE(34,115) (I,KDIAG(I),I=1,NF)
  115 FORMAT(o(2I7,3X))
  120 IF(NA .LE. MAXA)GO TO 140
C
      WRITE(31,130)NA,MAXA
  130 FORMAT('1**** ERROR CONCERNING MAXA *****',///,' THE SIZE OF MATRIX
     * A (NA) SHOULD BE ',I8,' , BUT MAXA IS ',I8)
      STOP
  140 RETURN
      END
```

SETUP

```
C
C*********************************************************************
C
C  THE SUBROUTINE SETUP GENERATES A LIST SHOWING THE RELATIONSHIP THE
C  NODES HAVE WITH EACH OTHER, AND COMPUTES THE ORIGINAL MATRIX BAND-
C  WIDTH (ADAPTED FROM COLLINS, 1973)
C
      SUBROUTINE SETUP(JT,JMEM,MEMJT)
      COMMON /AREA2/ MAXNOD,MAXELT,NODES,LMENTS,NC,NF,NA,MAXA,
     *MAXCON,MAXFRE,IPRT,JAP
      COMMON /AREA5/ MAXEL2,MAXN04,IDIFF
      INTEGER JT(MAXEL2),JMEM(MAXNOD),MEMJT(MAXN04)
C
      IDIFF=0
      DO 10 J=1,NODES
   10 JMEM(J)=0
      DO 60 J=1,LMENTS
      DO 50 I=1,2
      JNTI=JT(MAXELT*(I-1)+J)
      JSUB=(JNTI-1)*4
      DO 40 II=1,2
      IF(II.EQ.I)GOTO40
      JJT=JT(MAXELT*(II-1)+J)
      IF(JJT.EQ.0)GOTO50
      MEM1=JMEM(JNTI)
      IF(MEM1.EQ.0)GOTO30
      DO 20 III=1,MEM1
      IF(MEMJT(JSUB+III).EQ.JJT)GOTO40
   20 CONTINUE
   30 JMEM(JNTI)=JMEM(JNTI)+1
      MEMJT(JSUB+JMEM(JNTI))=JJT
      IF(IABS(JNTI-JJT).GT.IDIFF)IDIFF=IABS(JNTI-JJT)
   40 CONTINUE
   50 CONTINUE
   60 CONTINUE
      RETURN
      END
```

OPTNUM

```
C
C ***************************************************************
C
C  THE SUBROUTINE OPTNUM RENUMBERS THE NODES IN ORDER TO MINIMIZE
C  THE BAND-WIDTH OF THE MATRIX (MODIFIED FROM COLLINS, 1973).
C
      SUBROUTINE OPTNUM(JT,JMEM,MEMJT,NODNUM,NEWNOD,JOINT,NEWJT)
      COMMON /AREA2/ MAXNOD,MAXELT,NODES,LMENTS,NC,NF,NA,MAXA,
     *MAXCON,MAXFRE,IPRT,JAP
      COMMON /AREA5/ MAXEL2,MAXNO4,IDIFF
      INTEGER JOINT(MAXNOD),NEWJT(MAXNOD)
      INTEGER JT(MAXEL2),JMEM(MAXNOD),MEMJT(MAXNO4),NODNUM(MAXNOD),
     *NEWNOD(MAXNOD)
C
      IDIFF1=IDIFF
      MINMAX=IDIFF
      DO 60 IK=1,NODES
      DO 20 J=1,NODES
      JOINT(J)=0
   20 NEWJT(J)=0
      MAX=0
      I=1
      NEWJT(1)=IK
      JOINT(IK)=1
      K=1
   30 K4=JMEM(NEWJT(I))
      IF(K4.EQ.0)GOTO45
      JSUB=(NEWJT(I)-1)*4
      DO 40 JJ=1,K4
      K5=MEMJT(JSUB+JJ)
      IF(JOINT(K5).GT.0)GOTO40
      K=K+1
      NEWJT(K)=K5
      JOINT(K5)=K
      IDIFF=IABS(I-K)
      IF(IDIFF.GE.MINMAX)GOTO60
      IF(IDIFF.GT.MAX)MAX=IDIFF
   40 CONTINUE
      IF(K.EQ.NODES)GOTO50
      IF(K.GT.I)GOTO45
C
C IN CASE OF A DISCONNECTED NETWORK ....
C
           DO 42 J=1,NODES
           IF(JOINT(J).EQ.0)GOTO43
   42      CONTINUE
           GO TO 45
   43      K=K+1
           NEWJT(K)=J
           JOINT(J)=K
C
   45 I=I+1
      GOTO30
   50 MINMAX=MAX
```

```
 OPTNUM


C
      DO 55 J=1,NODES
      NEWNOD(J)=NEWJT(J)
   55 NODNUM(J)=JOINT(J)
   60 CONTINUE
C
      IF(IDIFF .LT. IDIFF1) GO TO 85
C
C IN CASE WHERE NO RENUMBERING IS REQUIRED
      DO 70 M=1,NODES
      NEWNOD(M)=M
70    NODNUM(M)=M
C PRINT OUT THE FINAL NODE NUMBERING
C
85    WRITE(31,90) IDIFF
   90 FORMAT(////,10X,'OUTPUT FROM OPTNUM',///,' FINAL IDIFF=',I5,///)
      IF(IPRT .LT. 3)GO TO 150
C
      WRITE(34,105)
  105 FORMAT(////,10X,'FINAL NODE RENUMBERING',///,
     *10X,'FORMAT(5(I  <==NEWNOD  ==>NODNUM))',///)
      WRITE(34,112)(I,NEWNOD(I),NODNUM(I) , I=1,NODES)
  112 FORMAT(5(3I5,5X))
C
  150 RETURN
      END
```

SOLPHI

```
C*********************************************************************
C
C  THE SUBROUTINE SOLPHI CALCULATES THE VALUES IN THE SUBMATRICES
C  A AND PHIC, AND SOLVES THE SYSTEM FOR THE UNKNOWN VALUES OF PHI
C  USING THE CHOLESKY METHOD.
C  SOLPHI CALLS THE SUBROUTINES DECOMP, FORW AND BACK.
C
      SUBROUTINE SOLPHI(IB,LC,IDIAG,NOD1,NOD2,XORD,YORD,PHI,PHIC,W,A,
     *E,NODNUM,NEWNOD,REOR,ELLEN)
      COMMON /AREA2/ MAXNOD,MAXELT,NUMNPT,NUMELT,NC,NF,NA,MAXA,
     *MAXCON,MAXFRE,IPRT,JAP
      COMMON /AREA4/ XB(8),YB(8),HDB(8),HDE(8),ISHAP(8),IBC(8),IGEOM,NBO
      COMMON /AREA6/ TITLE(20)
      COMMON /AREA8/ VISC,SPGR
      INTEGER IDIAG(MAXFRE)
      INTEGER IB(MAXNOD),LC(MAXNOD),NODNUM(MAXNOD),
     *NEWNOD(MAXNOD),NOD1(MAXELT),NOD2(MAXELT)
      DIMENSION XORD(MAXNOD),YORD(MAXNOD),PHI(MAXNOD)
      DIMENSION E(MAXELT),ELLEN(MAXELT),W(MAXELT)
      DIMENSION IBCO(2),INEW(2)
      DIMENSION PHIC(MAXCON)
      DIMENSION A(MAXA)
      LOGICAL REOR(MAXNOD)
C
C  CONSTANTS
      FLUID=SPGR/(12*VISC)
C
C  CLEAR ARRAY A
      DO 5 I=1,MAXA
    5 A(I)=0.0
      IF(JAP .GT. 1)GO TO 13
C
C  SET UP CONSTRAINED HEAD VECTOR (PHIC)
      DO 10 I=1,NUMNPT
        M9=NEWNOD(I)
        M11=IB(M9)
        IF(M11 .EQ. 0)GO TO 10
        IF(IBC(M11) .EQ. 0)GO TO 10
        PHIC(LC(I))=PHI(M9)
   10 CONTINUE
C
C  CLEAR ARRAY PHI
   13 DO 15 I=1,NF
   15 PHI(I)=0.0
C
C  CALCULATE THE COEFFICIENT OF PHI FOR EACH ELEMENT AND SET UP THE
C  SUBMATRIX A (FOR TOTALLY FREE ELEMENTS).
C
      DO 250 N=1,NUMELT
C
C  CALCULATE COEFFICIENT E FOR ELEMENT N
      IOLD=NOD1(N)
      IBO=IB(IOLD)
      IBCO(1)=0
      IF(IBO .EQ. 0)GO TO 18
```

```
      IBCO(1)=IBC(IBO)
   10 INEW(1)=NODNUM(IOLD)
C
      IOLD=NOD2(N)
      IBO=IB(IOLD)
      IBCO(2)=0
      IF(IBO .EQ. 0)GO TO 20
      IBCO(2)=IBC(IBO)
   20 INEW(2)=NODNUM(IOLD)
C
      E(N)=(W(N)**3)*FLUID / ELLEN(N)
C
C ASSIGN THE VALUE 0, OR 1, OR 2, TO IB2, IF 0 NODE, 1 NODE, OR BOTH
C NODES ARE CONSTRAINED, RESPECTIVELY.
      IF(IBCO(1) .EQ. 0 .AND. IBCO(2) .EQ. 0)GO TO 30
      IF(IBCO(1) .EQ. 0 .OR. IBCO(2) .EQ. 0)GO TO 25
      IB2=2
      GO TO 35
   25 IB2=1
      GO TO 35
   30 IB2=0
   35 CONTINUE
C
C INSERT E INTO THE GLOBAL COEFFICIENT SUBMATRICES A OR AC.
C
      II=MAX0(INEW(1),INEW(2))
      JJ=MIN0(INEW(1),INEW(2))
C
C SUBMATRIX A : LINEAR ARRAY AND VARIABLE BANDWIDTH STORAGE.
      DO 140 I=1,2
      IF(IBCO(I) .GT. 0)GO TO 140
      M4=KDIAG(INEW(I)-LC(INEW(I)))
      A(M4)=E(N)+A(M4)
  140 CONTINUE
      IF(IB2-1)160,170,250
  160 M5=KDIAG(II-LC(II)) - (II-JJ) + (LC(II)-LC(JJ))
      IF(M5 .LT. 1)GO TO 165
      A(M5)=-E(N)
      GO TO 250
  165 WRITE(31,166)M5
  166 FORMAT('1**** ERROR IN MATRIX A ********',///,' M5 = ',I6,' (???)')
      STOP
C
C AUGMENTED FLUX VECTOR (I.E. ENTRIES OF NODE CONDUCTANCE MATRIX FOR
C PARTIALLY CONSTRAINED ELEMENTS MULTIPLIED BY CORRESPONDING ENTRIES
C OF VECTOR PHIC )   STORED IN VECTOR PHI, WHICH HAS BEEN CLEARED
C PREVIOUSLY.
C
  170 DO 190 I=1,2
      IF(IBCO(I) .GT. 0)GO TO 190
      M6=INEW(I)-LC(INEW(I))
      II=IABS(I-3)
      M7=LC(INEW(II))
  190 CONTINUE
      AC=E(N)
      PHI(M6)=AC*PHIC(M7) + PHI(M6)
  250 CONTINUE
```

```
  SOLPHI

      IF(IPRT .LT. 3)GO TO 330
C
C PRINT OUT THE MATRICES TO BE SOLVED
C
      WRITE(34,295) JAP
  295 FORMAT(//////,'1 MATRICES A, ? (STORED IN ARRAY PHI), AND PHIC, BEF
     *ORE DECOMPOSITION',5X,'JAP=',I4,///,
     *5X,'MATRIX A',8X,'FORMAT(5(I    A(I)))',//)
      WRITE(34,301) (I,A(I),I=1,NA)
  301 FORMAT(5(4X,I3,1PE13.3))
C
      WRITE(34,313)JAP
  313 FORMAT(////9X,'VECTOR ''B'', STORED IN ARRAY PHI',10X,'FORMAT(5(I
     *  PHI(I)))',5X,'JAP=',I4,//)
      WRITE(34,316) (I,PHI(I),I=1,NF)
  316 FORMAT(5(I5,1PE12.3))
C
      WRITE(34,318) JAP
  318 FORMAT(////9X,'VECTOR PHIC',10X,'FORMAT(5(I    PHIC(I)))',5X,'JAP=',
     *I4,//)
      WRITE(34,321) (I,PHIC(I),I=1,NC)
  321 FORMAT(5(I5,1PE12.3))
C
C CHOLESKI DECOMPOSITION.
C
  330 CALL DECOMP(A,KDIAG)
C
C SOLVE FOR PHI, IN TWO STEPS:FORWARD-SUBSTITUTION AND BACKSUBSTITUTION
      CALL FORW(A,KDIAG,PHI)
C
      CALL BACK(A,KDIAG,PHI)
C
C EXPAND HEAD VECTOR
C
      DO 400 MM=1,NUMNPT
      M=NUMNPT-MM+1
      M10=IB(NEWNOD(M))
      IF(M10 .EQ. 0)GO TO 340
      IF(IBC(M10) .GT. 0)GO TO 350
  340 K=M-LC(M)
      PHI(M)=PHI(K)
      GO TO 400
  350 PHI(M)=PHIC(LC(M))
  400 CONTINUE
C
C REORDER PHI ARRAY IN INITIAL NODE ORDER.
C
      DO 410 M=1,NUMNPT
  410 REOR(M)=.FALSE.
      DO 450 M=1,NUMNPT
      IF(REOR(M))GO TO 450
      PHIM=PHI(M)
      MA=NEWNOD(M)
  420 PHIMA=PHI(MA)
      REOR(MA)=.TRUE.
      PHI(MA)=PHIM
      MA=NEWNOD(MA)
```

```
 SOLPHI

      IF(REOR(MA))GO TO 450
      PHIM=PHIMA
      GO TO 420
  450 CONTINUE
C
      IF(IPRT .LT. 2)GO TO 500
C
C PRINT THE RESULTS OF FLOW CALCULATION.
      WRITE(33,470) TITLE,JAP
470   FORMAT('1',/,1X,20A4,/,' FILE: FLO4LL.U33',
     +///,10X,'RESULTS OF FLOW CALCULATIONS,  JAP=',I4,///
     *10X,'HEAD AT NODAL POINTS',///,
     *10X,'FORMAT(3(I    IB(I)    PHI(I)))',///)
      WRITE(33,480)(M,IB(M),PHI(M), M=1,NUMNPT)
  480 FORMAT(4(I5,I4,1PE15.7,3X))
  500 RETURN
      END
```

```
C*********************************************************************
C
C  THE SUBROUTINE DECOMP OPERATES A CHOLESKI DECOMPOSITION ON A
C  VARIABLE BANDWIDTH STORAGE MATRIX (FROM JENNINGS, 1977)
C
      SUBROUTINE DECOMP(A,KDIAG)
      COMMON /AREA2/ MAXNOD,MAXELT,NUMNPT,NUMELT,NC,NF,NA,MAXA,
     *MAXCON,MAXFRE,IPRT,JAP
      INTEGER KDIAG(MAXFRE)
      DIMENSION A(MAXA)
C
      A(1)=SQRT(A(1))
      DO 1 I=2,NF
      KI=KDIAG(I)-I
      L=KDIAG(I-1)-KI+1
      DO 2 J=L,I
      X=A(KI+J)
      KJ=KDIAG(J)-J
      IF(J .EQ. 1)GO TO 2
      LBAR=KDIAG(J-1)-KJ+1
      LBAR=MAXO(L,LBAR)
      IF(LBAR .EQ. J)GO TO 2
      JJ=J-1
      DO 3 K=LBAR,JJ
    3 X=X-A(KI+K)*A(KJ+K)
    2 A(KI+J)=X/A(KJ+J)
      IF(X .LE. 0.0) THEN
        WRITE(31,20) I,X
   20   FORMAT(////,' ***** WARNING *****',//,' IN SUBROUTINE DECOMP, AT
     * I=',I5,',  X WAS :',1PE15.3,//,' THE SIGN OF X HAS BEEN CHANGED TO
     * ALLOW SQRT(X) TO BE COMPUTED.')
        X=-X
      END IF
    1 A(KI+I)=SQRT(X)
C
      RETURN
      END
```

FORW

```
C
C*************************************************************************
C
C THE SUBROUTINE FORW EXECUTES A FORWARD SUBSTITUTION IN WHICH THE
C VECTOR B IS OVERWRITTEN BY THE SOLUTION Y OF THE MATRIX EQUATION
C LY=B  (FROM JENNINGS, 1977).
C
      SUBROUTINE FORW(A,KDIAG,B)
      COMMON /AREA2/ MAXNOD,MAXELT,NUMNPT,NUMELT,NC,NF,NA,MAXA,
     *MAXCON,MAXFRE,IPRT,JAP
      INTEGER KDIAG(MAXFRE)
      DIMENSION A(MAXA),B(MAXNOD)
C
      B(1)=B(1)/A(1)
      DO 4 I=2,NF
      KI=KDIAG(I)-I
      L=KDIAG(I-1)-KI+1
      X=B(I)
      IF(L .EQ. I)GO TO 4
      II=I-1
      DO 5 J=L,II
    5 X=X-A(KI+J)*B(J)
    4 B(I)=X/A(KI+I)
C
      RETURN
      END
```

BACK

```
C
C*************************************************************************
C
C  THE SUBROUTINE BACK EXECUTES A BACKSUBSTITUTION IN WHICH THE VECTOR
C  Y, STORED IN B, IS OVERWRITTEN BY THE SOLUTION X OF THE MATRIX
C  EQUATION L(TRANSP)*X=Y (FROM JENNINGS, 1977).
C
      SUBROUTINE BACK(A,KDIAG,B)
      COMMON /AREA2/ MAXNOD,MAXELT,NUMNPT,NUMELT,NC,NF,NA,MAXA,
     *MAXCON,MAXFRE,IPRT,JAP
      INTEGER KDIAG(MAXFRE)
      DIMENSION A(MAXA),B(MAXNOD)
C
      DO 6 IT=2,NF
      I=NF+2-IT
      KI=KDIAG(I)-I
      X=B(I)/A(KI+I)
      B(I)=X
      L=KDIAG(I-1)-KI+1
      IF(L .EQ. I)GO TO 6
      II=I-1
      DO 7 K=L,II
    7 B(K)=B(K)-X*A(KI+K)
    6 CONTINUE
      B(1)=B(1)/A(1)
C
      RETURN
      END
```

FLOCAL

```
C
C*****************************************************************
C
C     THE SUBROUTINE FLOCAL CALCULATES THE FLOW IN EACH ELEMENT,
C   THE TOTAL FLOW THROUGH THE MODEL AND THE DIRECTIONAL
C   PARAMETERS
C
      SUBROUTINE FLOCAL(NOD1,NOD2,W,ELLEN,E,PHI,IB,ELOR)
      COMMON /AREA2/ MAXNOD,MAXELT,NUMNPT,NUMELT,NC,NF,NA,MAXA,
     *MAXCON,MAXFRE,IPRT,JAP
      COMMON /AREA4/ XB(8),YB(8),HDB(8),HDE(8),ISHAP(8),IBC(8),IGEOM,N30
      COMMON /AREA8/ VISC,SPGR
      COMMON /AREA9/ DVEL(36),DFR(36),DLE(36),DLVEL(36),DLVEL2(36),
     +DPOR(18),DK(18),DLE180(18),NSEG(36)
      INTEGER IB(NUMNPT)
      INTEGER NOD1(NUMELT),NOD2(NUMELT)
      DIMENSION M2(2),DINOUT(8)
      DIMENSION W(NUMELT),ELLEN(NUMELT),E(NUMELT),ELOR(NUMELT)
      DIMENSION PHI(NUMNPT)
C
C CLEAR ARRAYS
      DO 15 I=1,NBC
   15 DINOUT(I)=0.C
      DO 20 I=1,36
        DVEL(I)=0.C
        DFR(I)=0.C
        DLE(I)=0.0
        DLVEL(I)=0.0
        DLVEL2(I)=C.0
        NSEG(I)=0
   20 CONTINUE
      DO 24 I=1,18
      DPOR(I)=0.0
      DK(I)=0.0
      DLE180(I)=0.C
24    CONTINUE
C
C PRINT HEADINGS ON DETAILED OUTPUT FILES
      IF(IPRT .LT. 2)GO TO 48
      WRITE(33,45) JAP
   45 FORMAT(////,10X,'FLOW IN ELEMENTS,     JAP=',I4,
     *///,' NUMBER',3X,'WIDTH(M)',6X,'LENGTH(M)',4X,'VELOC (M/SEC)',
     +' FL.RATE(M**3/SEC) REYNOLD NO.    DIRECTION    TRANSIT TIME',//)
   48 WRITE(31,50)
50    FORMAT('1',//,10X,'SELECTED RESULTS OF FLOW CALCULATIONS',10X,
     *' FOR SEGMENTS LOCATED AT A BOUNDARY',///,' BDRY.NO   ELEM.NO
     *      WIDTH(M)            LENGTH(M)      VELOCITY(M/SEC)    FLOW RA
     *TE(M**3/SEC)',///)
      DO 100 N=1,NUMELT
C
C CALCULATE FLOW RATE,VELOCITY AND REYNOLD NUMBER IN EACH ELEMENT
      M2(1)=NOD1(N)
      M2(2)=NOD2(N)
      THN=ELOR(N)
      WN=W(N)
```

```
FLOCAL

      ELLENN=ELLEN(N)
      FR=E(N)*(PHI(M2(1))-PHI(M2(2)))
      VEL=FR/WN
      RE=ABS(FR/VISC)
      TRTIM=ABS(ELLENN/VEL)
      IF(IPRT .LT. 2)GO TO 55
      WRITE(33,53)N,WN,ELLENN,VEL,FR,RE,THN,TRTIM
   53 FORMAT(I5,1P7E15.7)
   55 DO 60 I=1,2
        M3=I3(M2(I))
        IF(M3 .EQ. 0)GO TO 60
        M1=I3C(M3)
        IF(M1 .EQ. 0)GO TO 60
        FR1=FR
        IF(I .EQ. 2) FR1=-FR
        WRITE(31,58) M3,N,WN,ELLENN,VEL,FR1
   58   FORMAT(2(I5,5X),1P4E20.7)
        DINOUT(M3)=DINOUT(M3) + FR1
   60 CONTINUE
C
C COMPUTE DIRECTIONAL POROS. & PERM. (0-180 DEGREES)
      IDIR=INT(THN/10.) + 1
      DPOR(IDIR)=DPOR(IDIR) + WN*ELLENN
      DK(IDIR)=DK(IDIR) + WN*WN*ELLENN
      DLE180(IDIR)=DLE180(IDIR) + ELLENN
C COMPUTE OTHER DIRECTIONAL PARAMETERS (0-360 DEGREES)
      IF(FR .LE. 0.0) THN=THN+180.
      IDIR=INT(THN/10.)+1
      AVEL=ABS(VEL)
      AFR=ABS(FR)
      DVEL(IDIR)=DVEL(IDIR)+AVEL
      DFR(IDIR)=DFR(IDIR)+AFR
      DLE(IDIR)=DLE(IDIR)+ELLENN
      DLVEL(IDIR)=DLVEL(IDIR)+ELLENN*AVEL
      DLVEL2(IDIR)=DLVEL2(IDIR) + ELLENN*AVEL*AVEL
      NSEG(IDIR)=NSEG(IDIR)+1
  100 CONTINUE
C
C PRINT SUMMARY OUTPUT FILES
      CALL OUTPT4(DINOUT)
      RETURN
      END
```

```
C
C*********************************************************************
C
      SUBROUTINE OUTPT4(DINOUT)
C THIS SUBROUTINE PRINTS OUT A SUMMARY OF FLOW CALCULATIONS
C AND VALUES OF DIRECTIONAL PARAMETERS
C
      COMMON /AREA2/ MAXNOD,MAXELT,NUMNPT,NUMELT,NC,NF,NA,MAXA,
     *MAXCON,MAXFRE,IPRT,JAP
      COMMON /AREA4/ X9(8),Y9(8),NDB(8),NDE(8),ISHAP(8),IBC(8),IGEOM,NBO
      COMMON /AREA9/ DVEL(36),DFR(36),DLE(36),DLVEL(36),DLVEL2(36),
     *DPOR(18),DK(18),DLE180(18),NSEG(36)
      DIMENSION DINOUT(8)
      CHARACTER BCD*26
C
C PRINT SUMMARY OF FLOW CALCULATIONS
      WRITE(31,110) JAP
  110 FORMAT('1',20X,'SUMMARY OF FLOW CALCULATIONS FOR JAP=',I4,
     */////,' TOTAL FLOW ALONG THE BOUNDARIES',///,
     *' BDRY NO',5X,'FLOW (M**3/SEC) (- =>FLOW OUT)'//)
      WRITE(31,115) (I,DINOUT(I), I=1,NBO)
  115 FORMAT(I5,1PE20.10)
      IF(IGEOM .EQ. 1) GO TO 125
      DELTAFR=DINOUT(3)+DINOUT(6)
      AVGFR=(DINOUT(3)-DINOUT(6))/2.0
      EORFR=DELTAFR/AVGFR
      WRITE(31,117) DELTAFR,AVGFR,EORFR
  117 FORMAT(/,2X,'DIFFERENCE IN FLOW RATE BETWEEN UP AND DOWNSTREAM',
     */,14X,'BOUNDARIES  3  AND  7 ',//,10X,'DELTA(FL.RA)',11X,
     *'AVG(FL.RA)',7X,'RELATIVE ERROR(FL.RA)',//,5X,
     *3(8X,1P1E12.3))
C
C PRINT DIRECTIONAL PARAMETERS
125   WRITE(31,126)
126   FORMAT(////,10X,'DIRECTIONAL POROSITY AND PERMEABILITY',///,
     *' DIRECTION          LENGTH        POROSITY    PERMEABILITY',//,
     *' (DEGR/10)          (M)             (M**2)        (M**3)',///)
      WRITE(31,129) (I,DLE180(I),DPOR(I),DK(I),I=1,18)
129   FORMAT(I5,5X,1P3E15.2)
      WRITE(31,130)
130   FORMAT('1',10X,'OTHER DIRECTIONAL PARAMETERS',///,
     *' DIRECTION       NSEG  (1)VELOC  (2)FL.RA  (3)LENGTH',
     *'(3)X(1) (3)X(1)**2',//,
     *' (DEGR/10)    ',10X,'(M/S)     (M**3/S)',7X,'(M)',///)
      WRITE(31,135) (I,NSEG(I),DVEL(I),DFR(I),DLE(I),
     *DLVEL(I),DLVEL2(I),I=1,36)
  135 FORMAT(I5,5X,I10,1P5E10.3)
      WRITE(32) DVEL,DLVEL,DLVEL2,DFR,DLE,NSEG
C
C CONSTRUCT ROSE DIAGRAMS
      IF(IPRT-2) 150,140,138
138   BCD='VELOCITIES (M/S)'
      CALL ROSE(DVEL,BCD)
      BCD='SEGMENT LENGTHS (M)'
      CALL ROSE(DLE,BCD)
```

```
OUTPT4

      BCD='LENGTH X VELOC**2'
      CALL ROSE(DLVEL2,BCD)
140   BCD='FLOW RATES (M**3/S)'
      CALL ROSE(DFR,BCD)
  150 CONTINUE
      RETURN
      END
```

```
C*************************************************************************
C                                                                        *
C       THIS SUBROUTINE PRINTS A ROSE DIAGRAM FOR A DIRECTIONAL          *
C     PARAMETER                                                          *
C                                                                        *
C*************************************************************************
      SUBROUTINE ROSE(R2,BCD1)
      DIMENSION R(36),R2(36),COSTH(36),SINTH(36)
      CHARACTER LPLOT(80,64)*1,BCD1*26,N1*1,N2*1,N3*1
      COMMON /AREA12/ LPLOT
      N1='.'
      N2='0'
      N3='*'
      COUNT=0.0
      IBRANCH=1
      DO 10 J=1,64
        DO 5 I=1,79,2
          LPLOT(I,J)=' '
          LPLOT(I+1,J)=' '
    5   CONTINUE
   10 CONTINUE
C
C COMPUTE THE ARRAYS COSTH() AND SINTH()
      DO 20 I=1,36
        IDEG=I*10
        FDEG=FLOAT(IDEG)-0.5
        COSTH(I)=COSD(FDEG)
        SINTH(I)=SIND(FDEG)
   20 CONTINUE
C
      WRITE(31,25) BCD1
   25 FORMAT('1',//,'T',//,20X,'ROSE OF ',A,//)
      CALL MXMNAS(R2,36,RMAX,RMIN,AVG,SIGMA,25)
C
C    DRAW A CIRCLE WITH A MARK EVERY 10 DEGREES
      DO 30 I=1,36
   30 CALL MARK(COSTH(I)*.95,SINTH(I)*.95,N1)
C
C    SCALE ARRAYS
      DO 100 I=1,36
        R(I)=R2(I)/(RMAX*1.05)
        IF(R(I).LT..01) R(I)=.01
        IF(R(I).GT.0.1) COUNT=COUNT+1.0
  100 CONTINUE
C
C    FOR COUNT LESS THAN 3 USE LOG VALUES
      IF(COUNT.LT.3.) IBRANCH=2
      DO 300 I=1,36
        IF(R(I).LT.0.02) GOTO 300
        IF(I.EQ.1) THEN
          C1=1.0
          S1=0.0
        ELSE
          C1=COSTH(I-1)
          S1=SINTH(I-1)
```

```
      ENDIF
      GOTO(150,160) IBRANCH
  150 X1=R(I)*C1
      Y1=R(I)*S1
      X2=R(I)*COSTH(I)
      Y2=R(I)*SINTH(I)
      GOTO 200
  160 X1=(C1*LOG10(R(I)*100))/2.0
      Y1=(S1*LOG10(R(I)*100))/2.0
      X2=(COSTH(I)*LOG10(R(I)*100))/2.0
      Y2=(SINTH(I)*LOG10(R(I)*100))/2.0
C
C   DRAW A LINE BETWEEN THE TWO POINTS
  200 CALL MARK(X1,Y1,N3)
      CALL MARK(X2,Y2,N3)
      DX=(X1-X2)/5.0
      DY=(Y1-Y2)/5.0
      RINC=0.0
  225 RINC=RINC+1
      XM=X2+DX*RINC
      YM=Y2+DY*RINC
      CALL MARK(XM,YM,N3)
      IF(RINC.LT.4.0) GOTO 225
  300 CONTINUE
      CALL MARK(0.,0.,N2)
      IF(IBRANCH.EQ.2) THEN
        WRITE(31,40)
   40   FORMAT(20X,'ROSE OF LOG VALUES',//)
      ENDIF
      DO 400 J=64,1,-1
        WRITE(31,80) (LPLOT(I,J),I=1,80)
   80   FORMAT(1X,80A1)
  400 CONTINUE
      RETURN
      END
```

MXMNAS

```
C*************************************************************************
C                                                                        *
C        THIS SUBROUTINE CALCULATES THE MAXIMUM,MINIMUM,AVERAGE AND      *
C      STANDARD DEVIATION FOR A SET OF DATA                              *
C                                                                        *
C*************************************************************************
      SUBROUTINE MXMNAS(ARRAY,NUM,XMAX,XMIN,TAVG,SIGMA,L)
      DIMENSION ARRAY(NUM)
C
C  INITIALIZE VARIABLES
      TTOTAL=ARRAY(1)
      T2TOTAL=ARRAY(1)*ARRAY(1)
      XMAX=ARRAY(1)
      XMIN=ARRAY(1)
      RNUM=FLOAT(NUM)
      DO 100 I=2,NUM
        IF(ARRAY(I).LT.XMIN) XMIN=ARRAY(I)
        IF(ARRAY(I).GT.XMAX) XMAX=ARRAY(I)
        TTOTAL=TTOTAL+ARRAY(I)
        T2TOTAL=T2TOTAL+ARRAY(I)*ARRAY(I)
  100 CONTINUE
      TAVG=TTOTAL/RNUM
      T2AVG=T2TOTAL/RNUM
      SIGMA=SQRT(T2AVG-(TAVG*TAVG))
      IF(L.GT.0) THEN
      WRITE(L,6)
    6 FORMAT(23X,'BASIC DESCRIPTIVE STATISTICS',//,5X,'NUMBER OF',
     *15X,'STANDARD',//,4X,'DATA POINTS     MEAN       DEVIATION',
     *4X,'MAXIMUM      MINIMUM',/)
      WRITE(L,7) NUM,TAVG,SIGMA,XMAX,XMIN
    7 FORMAT(7X,I4,4X,4(1PE11.3,1X),///)
      ENDIF
      RETURN
      END
```

MARK

```
C
C*********************************************************************
C
C     THE SUBROUTINE MARK INSERTS A MARK INTO THE LOCATION (X,Y).
C   THE MARK USED IS THAT DEFINED BY THE CHARACTER NH. NOTE THAT
C   BOTH X AND Y MUST BOTH BE IN THE RANGE OF -1 TO +1.
C
      SUBROUTINE MARK(X,Y,NH)
      COMMON/AREA12/ LPLOT
      CHARACTER NH*1,LPLOT(80,54)*1
C
C   SCALE X AND Y
      IX=INT((X*40)+40.5)
      IY=INT((Y*32)+32.5)
      IF(IX.LT.1) IX=1
      IF(IY.LT.1) IX=1
      LPLOT(IX,IY)=NH
      RETURN
      END
```

G.4 Listing of the Program NETRANS

NETRANS

```
      PROGRAM NETRANS
C*****************************************************************************
C
C   PROGRAM TO SIMULATE STOCHASTICALLY THE MIGRATION OF PARTICLES
C   THROUGH A FRACTURE NETWORK BASED ON STATISTICS OF THE DIRECTIONAL
C   PARAMETERS
C
C WRITTEN BY A. ROULEAU AND D. LENTZ, 1984; UPDATE BY A.ROULEAU, JANUARY 1986.
C
C*****************************************************************************
      COMMON /AREA1/ COSTH(36),SINTH(36),DLM(36),DWTV(36),RDFR(36),
     +XLEN,YLEN,TRAVDIS,IDIR(100),MPART(10),IBC1,IPRT,JAP
      COMMON/AREA2/TITLE
      DIMENSION DVEL(36),DFR(36),DLE(36),DLVEL(36),DLVEL2(36),NSEG(36)
      CHARACTER TITLE(20)*4,FMT(5)*20
      DATA (FMT(I),I=1,5)/'(2(/),10I3)','(4(/),10I3)','(6(/),10I3)',
     +'(8(/),10I3)','(10(/),10I3)'/
      MAXNPRT=50
C
C   READ IN INPUT DATA
      READ(1,5)TITLE
    5 FORMAT(20A4)
      WRITE(41,6) TITLE
6     FORMAT('1',//,1X,20A4,//,' FILE : TRANSIT.U41')
      READ(1,10) NSET,IGEOM,NAP,NAAP,IPRT,NPART,TRAVDIS,NMAP
   10 FORMAT(I5,20X,3I5,10X,I5,I5,F5.0,I3)
      NAPER=NAP*NAAP
      IF(IGEOM .EQ. 2) THEN
        WRITE(41,12)
12      FORMAT(///,' ****  SORRY ! *****',///,10X,
     +  'THIS VERSION OF NETRANS DOES NOT APPLY TO CIRCULAR GEOMETRY')
        STOP
      ENDIF
      IF(NPART.GT.MAXNPRT) THEN
        WRITE(41,15) NPART,MAXNPRT
   15   FORMAT('1',10X,'***ERROR AT INPUT STAGE***',///,10X,
     +  'NPART=',I5,' ,  THIS IS LARGER THAN MAXNPRT=',I5)
        STOP
      ENDIF
      READ(1,20) XMINI,YMAXI,XMAXI,YMINI
   20 FORMAT(20X,4F5.0)
      XLEN=XMAXI-XMINI
      YLEN=YMAXI-YMINI
      READ(1,40) IBC1
   40 FORMAT(20X,I5,7(/))
      IF(NMAP.GT.0) THEN
        READ(1,FMT(NSET))(MPART(I),I=1,NMAP)
      ENDIF
        DO 70 I=1,36
          IDEG=I*10
          FDEG=FLOAT(IDEG)-5.0
          COSTH(I)=COSD(FDEG)
          SINTH(I)=SIND(FDEG)
   70   CONTINUE
C
```

```
C  READ IN DIRECTIONAL PARAMETERS
      DO 400 JAP=1,NAPER
        READ(32) DVEL,DLVEL,DLVEL2,DFR,DLE,NSEG
C
C  COMPUTE AUXILIARY STATISTICS OF DIRECTIONAL PARAMETERS
        SUMFR=0.0
        DO 50 I=1,36
   50   SUMFR=SUMFR+DFR(I)
        IF(SUMFR.EQ.0.0)THEN
          WRITE(41,56)
56        FORMAT(////,1X,'SUMFR=0.0 : ERROR PROBABLY IN INPUT ',
     +    'DIRECTIONAL PARAMETERS')
          STOP
        ENDIF
        DO 60 I=1,36
          IF(NSEG(I).EQ.0) THEN
            RDFR(I)=0.0
            DLM(I)=0.0
          ELSE
            RDFR(I)=DFR(I)/SUMFR
            DLM(I)=DLE(I)/NSEG(I)
            DWTVM(I)=DLVEL2(I) / DLVEL(I)
          ENDIF
   60   CONTINUE
C
C COMPUTE IDIR(J), THE DIRECTION (1 TO 36) A PARTICLE SHALL
C TAKE FOR CORRESPONDING VALUE OF RANDOM J (1 TO 100)
        DO 71 I=1,100
71      IDIR(I)=0
        IR=1
        DO 100 I=1,36
          IF (RDFR(I).LE.0.005) GOTO 100
          R100=RDFR(I)*100.0
          IR100=NINT(R100)
          IREND=IR+IR100-1
          IF(IREND.GT.100) IREND=100
          DO 90 J=IR,IREND
   90     IDIR(J)=I
          IR=IREND+1
  100   CONTINUE
C
C PRINT OUT PARTICLE TRANSIT PARAMETERS
        CALL OUTPTS
C
C  COMPUTE TRANSIT TIME OF PARTICLES THROUGH THE NETWORK
        CALL TTIMER(NPART,NMAP)
  400 CONTINUE
      STOP
      END
```

```
C
C ******************************************************************
C
      SUBROUTINE OUTPTS
C
C THIS SUBROUTINE PRINTS OUT INFORMATIONS ON PARTICLE TRANSIT
C PARAMETERS.
C
      COMMON /AREA1/ COSTH(36),SINTH(36),DLM(36),DWTVM(36),RDFR(36),
     +XLEN,YLEN,TRAVDIS,IDIR(100),MPART(10),IBC1,IPRT,JAP
      CHARACTER BCD1*4
C
      WRITE(41,10) JAP
10    FORMAT(////,10X,'AUXILIARY DIRECTIONAL PARAMETERS FOR JAP=',I4)
      WRITE(41,20)
  20  FORMAT(/,1X,'DIRECTION',5X,'RDFR',7X,'DLM',7X,'DWTVM',//)
      WRITE(41,30) (I,RDFR(I),DLM(I),DWTVM(I),I=1,36)
  30  FORMAT(2X,I5,5X,1P3E10.2)
C
      WRITE(41,40)
40    FORMAT(/////,10X,'ARRAY IDIR',//,10X,'FORMAT: 10(J  IDIR(J))',///)
      WRITE(41,50) (J,IDIR(J), J=1,100)
50    FORMAT(10(3X,2(I4)))
C
      IF(IBC1 .EQ. 1) THEN
        BCD1=' X '
      ELSE
        BCD1=' Y '
      ENDIF
      WRITE(41,60) XLEN,YLEN,TRAVDIS,BCD1
60    FORMAT(///,10X,'SIZE OF GENERATED NETWORK :',//,15X,'X-LENGTH=',
     +F10.4,' (M)',//,15X,'Y-LENGTH=',F10.4,' (M)',///,5X,
     +'PARTICLES MUST TRAVEL ',F6.2,' (M) IN THE ',A4,' DIRECTION')
C
      RETURN
      END
```

```
C*****************************************************************************
C
C      SUBROUTINE TTIMER COMPUTES STOCHASTICALLY THE TRANSIT TIME OF
C  PARTICLES THROUGH THE NETWORK USING THE STATISTICS OF THE
C  DIRECTIONAL PARAMETERS (RECTANGULAR MODEL)
C
C*****************************************************************************
      SUBROUTINE TTIMER(NPART,NM)
      COMMON /AREA1/ COSTH(36),SINTH(36),DI 1(36),DWTVM(36),RDFR(36),
     +XLEN,YLEN,TRAVDIS,IDIR(100),MPART(10),IBC1,IPRT,JAP
      COMMON/AREA2/TITLE
      DIMENSION R(100),XTRAV(50),YTRAV(50),TIM(50)
     *,NSTEP(50),TRAV(50)
      DOUBLE PRECISION DSEED
      CHARACTER TITLE(20)*4,BCD1*22
C
      NR=100
      DSEED=1.D0
      ICOUNT=1
      NTEST=0
      NMAP=NM
      IF(NMAP.GT.0) THEN
        WRITE(41,20)
20      FORMAT(//,5X,'TRACKING OF SELECTED PARTICLES',//,6X,
     *  ' STEP',6X,'CUMX',6X,'CUMY',6X,'CUMT',/)
      ENDIF
C
C  COMPUTE TRAVEL TIME FOR NPART PARTICLES
      DO 150 NT=1,NPART
        CUMX=0.0
        CUMY=0.0
        CUMT=0.0
        CUML=0.0
        NSTEP(NT)=0
        IF(NMAP.GT.0) THEN
          IF(NT.EQ.MPART(ICOUNT)) THEN
            ICOUNT=ICOUNT+1
            IF(ICOUNT.GT.NMAP) THEN
              NMAP=0
            ENDIF
          NTEST=NT
          WRITE(41,35) NT
35        FORMAT(/,13X,'PARTICLE NUMBER ',I4,/,)
          ENDIF
        ENDIF
25      CALL GGUBS(DSEED,NR,R)
        DO 90 I=1,100
C
C  DETERMINE DIRECTION CORRESPONDING TO THE RANDOM NUMBER R(I)
          R100=R(I)*100.0
          IR100=INT(R100) +1
          JDIR=IDIR(IR100)
C
C  COMPUTE TRAVELLED DISTANCE AND TRAVEL TIME
          DL=DLM(JDIR)
```

```
            DX=DL*COSTH(JDIR)
            DY=DL*SINTH(JDIR)
            DT=DL/DWTVM(JDIR)
            CUMX=CUMX+DX
            CUMY=CUMY+DY
            CUMT=CUMT+DT
            CUML=CUML+DL
            IF(NT.EQ.NTEST) THEN
              NSTP=I+NSTEP(NT)*100
              WRITE(41,45) NSTP,CUMX,CUMY,CUMT
   45         FORMAT(7X,I3,3X,1P3E10.2)
            ENDIF
C
C CHECK IF TRAVELLED DISTANCE GREATER THAN TRAVDIS
            IF(IBC1.EQ.1)THEN
              IF(ABS(CUMX).GE.TRAVDIS)THEN
                GOTO 120
              ENDIF
            ELSE
              IF(ABS(CUMY).GE.TRAVDIS)THEN
                GOTO 120
              ENDIF
            ENDIF
   90     CONTINUE
            NSTEP(NT)=NSTEP(NT)+1
          GOTO 25
C
C TRIM THE TRAVELLED PATH AT LIMIT OF SYSTEM
120       IF(IBC1 .EQ. 1) THEN
            XOUT=ABS(CUMX)-TRAVDIS
            XOUT=SIGN(XOUT,CUMX)
            DOUT=XOUT/COSTH(JDIR)
            YOUT=DOUT*SINTH(JDIR)
          ELSE
            YOUT=ABS(CUMY)-TRAVDIS
            YOUT=SIGN(YOUT,CUMY)
            DOUT=YOUT/SINTH(JDIR)
            XOUT=DOUT*COSTH(JDIR)
          ENDIF
          XTRAV(NT)=CUMX-XOUT
          YTRAV(NT)=CUMY-YOUT
          TRAV(NT)=CUML-DOUT
          TIM(NT)=CUMT-DOUT/DWTVM(JDIR)
          NSTEP(NT)=NSTEP(NT)*100+I
  150   CONTINUE
C
C PRINT STATISTICS OF ALL PARTICLES
        WRITE(41,201) JAP
201     FORMAT('1',///,10X,'STATISTICS OF ALL PARTICLES FOR JAP=',I4,///
       +,5X,'NT',10X,'XTRAV(M)  YTRAV(M)   TRAV(M)   TIME(S)',5X,
       +'NSTEP',//)
        WRITE(41,202)(I,XTRAV(I),YTRAV(I),TRAV(I),TIM(I),NSTEP(I),
       +I=1,NPART)
  202   FORMAT(2X,I5,3X,1P4E10.2,2X,I6)
C
C COMPUTE BASIC STATISTICS AND MAKE FREQUENCY DIAGRAMS OF RESULTS
        ICD1='ABS.TRAV.DIST.,TRAV(M)'
```

```
TTIMER

      CALL MXYMAS(TPAV,NPART,XMAX,XMIN,TAVG,SIGMA,41,BCD1)
      BCD1='TRANSIT TIME (S)'
      CALL HSTGR(TIM,NPART,BCD1)
      RETURN
      END
```

```
C
C*************************************************************************
C
      SUBROUTINE HSTGR(ARRAY,NUM,T1)
C
C THIS SUBROUTINE PLOTS HORIZONTAL-BAR FREQUENCY HISTOGRAMS OF THE
C THE ARRAY ARRAY.
C
      DIMENSION ARRAY(NUM),FREQ(21)
      CHARACTER T1*22,A(101)*1
C
C CALCULATE BASIC STATISTICS AND PRINT HEADINGS
      WRITE(61,5) T1
5     FORMAT('1',15X,'FREQUENCY DIAGRAMS OF  ',A,//)
      CALL MXMNAS(ARRAY,NUM,XMAX,XMIN,TAVG,SIGMA,61,T1)
      WRITE(61,10)
   10 FORMAT(/,1X,'MIDDLE    FREQ  CUM REL',13X,'RELATIVE',
     *' FREQUENCY DIAGRAM',24X,'CUMULATIVE FREQUENCY',
     +' DIAGRAM',//,1X,'INTERV',9X,'FREQ',//)
C
      RANGE=XMAX-XMIN
      XINT=RANGE/20.0
      YC=0.0
      I=1
      C1=50.0/FLOAT(NUM)
      DO 50 I=1,21
   50 FREQ(I)=0.0
C
C DISPATCH ENTRIES INTO APPROPRIATE RANGE
      DO 150 I=1,NUM
        VAR=ARRAY(I)
        N=INT((VAR-XMIN)/XINT)+1
        FREQ(N)=FREQ(N)+1.0
  150 CONTINUE
      DO 300 I=1,21
        X=XMIN+XINT*(FLOAT(I)-.5)
        LY=NINT(FREQ(I))
        YC=YC+FREQ(I)*C1
        LYC=NINT(YC)
C
C  CREATE PRINTING ARRAY
        DO 250 IT=1,101
          ITEST=NINT(FREQ(I)*C1)
          IF(IT.GT.51)ITEST=(LYC+51)
          IF(IT.LE.ITEST) THEN
            A(IT)='*'
          ELSE
            A(IT)=' '
          ENDIF
  250   CONTINUE
C
C  PRINT DIAGRAMS
        A(51)='1'
        WRITE(61,225) X,LY,(LYC+LYC),(A(J),J=1,101)
  225   FORMAT(1X,1PE8.2,1X,I3,1X,I5,3X,'1',101A)
```

```
HSTGR

300 CONTINUE
      RETURN
      END
```

```
*XMNAS

C*****************************************************************************
C                                                                              *
C        THIS SUBROUTINE CALCULATES THE MAXIMUM,MINIMUM,AVERAGE AND            *
C      STANDARD DEVIATION FOR A SET OF DATA                                    *
C                                                                              *
C*****************************************************************************
      SUBROUTINE MXMNAS(ARRAY,NUM,XMAX,XMIN,TAVG,SIGMA,L,T1)
      DIMENSION ARRAY(NUM)
      CHARACTER T1*22
C
C   INITIALIZE VARIABLES
      TTOTAL=ARRAY(1)
      T2TOTAL=ARRAY(1)*ARRAY(1)
      XMAX=ARRAY(1)
      XMIN=ARRAY(1)
      RNUM=FLOAT(NUM)
      DO 100 I=2,NUM
        IF(ARRAY(I).LT.XMIN) XMIN=ARRAY(I)
        IF(ARRAY(I).GT.XMAX) XMAX=ARRAY(I)
        TTOTAL=TTOTAL+ARRAY(I)
        T2TOTAL=T2TOTAL+ARRAY(I)*ARRAY(I)
  100 CONTINUE
      TAVG=TTOTAL/RNUM
      T2AVG=T2TOTAL/RNUM
      SIGMA=SQRT(T2AVG-(TAVG*TAVG))
      IF(L.GT.0) THEN
      WRITE(L,5) T1
    5 FORMAT(/,1X,'BASIC DESCRIPTIVE STATISTICS OF ',A,//,
     +5X,'NUMBER OF',15X,'STANDARD',//,4X,'DATA POINTS     MEAN',
     +6X,'DEVIATION',4X,'MAXIMUM      MINIMUM',/)
      WRITE(L,7) NUM,TAVG,SIGMA,XMAX,XMIN
    7 FORMAT(7X,I4,4X,4(1PE11.3,1X))
      ENDIF
      RETURN
      END
```